Probabilistic Feature-Based Registration for Interventional Medicine

by

Seth D. Billings

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

August, 2015

Abstract

The need to compute accurate spatial alignment between multiple representations of patient anatomy is a problem that is fundamental to many applications in computer-integrated interventional medicine. One class of methods for computing such alignments is feature-based registration, which aligns geometric information of the shapes being registered, such as salient landmarks or models of shape surfaces. A popular algorithm for surface-based registration is the Iterative Closest Point (ICP) algorithm, which treats one shape as a cloud of points that is registered to a second shape by iterating between point-correspondence and point-registration phases until convergence.

In this dissertation, a class of "most likely point" variants on the ICP algorithm is developed that offers several advantages over ICP, such as high registration accuracy and the ability to confidently assess the quality of a registration outcome. The proposed algorithms are based on a probabilistic interpretation of the registration problem, wherein the point-correspondence and point-registration phases optimize the probability of shape alignment based on feature uncertainty models rather than

minimizing the Euclidean distance between the shapes as in ICP. This probabilistic framework is used to model anisotropic errors in the shape measurements and to provide a natural context for incorporating oriented-point data into the registration problem, such as shape surface normals. The proposed algorithms are evaluated through a range of simulation-, phantom-, and clinical-based studies, which demonstrate significant improvement in registration outcomes relative to ICP and state-of-the-art methods.

ABSTRACT

Primary Advisor	Professor Russell H. Taylor Department of Computer Science Johns Hopkins University
Co-Advisor	Assistant Professor Emad M. Boctor Department of Radiology Division of Medical Imaging Physics Johns Hopkins Medical Institutions
Readers	Professor Gregory D. Hager Department of Computer Science Johns Hopkins University
	Research Professor Peter Kazanzides Department of Computer Science Johns Hopkins University

Dedication

In dedication to my parents,

Dan and Susan Billings,

who gave wings to my early dreams

and established my confidence to embark on life's pursuits.

"For it is God which worketh in you

both to will and to do of His good pleasure."

—Philippians 2:13

Acknowledgments

To all who had a hand in this venture, whether directly or indirectly, I owe my deepest gratitude.

Firstly, I give my sincerest thanks to my advisor, Dr. Russell Taylor, for his consistent and indispensable guidance throughout my research, without which this dissertation would not have been possible. Many key contributions of this work follow from his unique insights and immense knowledge. Sincerest thanks are also due to my co-advisor, Dr. Emad Boctor, who first interested me in the topic of this dissertation and who provided key guidance and support along the way. In addition to my advisors, I would like to sincerely thank the other members of my thesis committee, Dr. Greg Hager and Dr. Peter Kazanzides, for their added guidance while targeting specific clinical applications for this work and for welcoming my involvement alongside their research efforts. I would also like to thank my thesis committee for their insightful critiques, which have greatly enhanced the content and clarity of this manuscript.

I am deeply grateful for the primary funding that I received from the National

ACKNOWLEDGMENTS

Science Foundation Graduate Research Fellowship Program, the National Institutes of Health Individual Graduate Partnership Program, and Equinox Corporation; their support made this work possible. Thanks are also due to Johns Hopkins University, Intuitive Surgical, Inc., and THINK Surgical, Inc. for additional funding and equipment support.

I would like to express special thanks to my mentors at the National Institutes of Health: to Dr. Brad Wood for the opportunity to collaborate within his lab and for partial funding of my research, to Dr. Ankur Kapoor for navigating my first steps through the maze of literature regarding registration research, and to Dr. Sheng Xu for his ever willing advice and encouragement.

I would like to thank my colleagues within the Laboratory for Computational Sensing and Robotics, of which there are too many to name them all, who have enriched my graduate experience through their friendships, helpfulness, stimulating conversations, and acts of kindness. I would like to thank Anton Deguet for his enduring patience and expert help in regards to using the CISST software libraries. Many thanks to Hyun Jae Kang for his aid in the use and customization of software for the lab's ultrasound platforms, to Alexis Cheng, Xiaoyu Guo, and Fereshteh Aalamifar for their expert assistance in calibrating tracked ultrasound probes, and to Dr. Simon Leonard, Dr. Austin Reiter, Dr. Masaru Ishii, and Ayushi Sinha for their important contributions in obtaining the clinical data used for the endonasal skull base surgery experiments.

ACKNOWLEDGMENTS

I would like to thank the exceptional staff of the computer science department and of the LCSR lab for making the administrative aspects of my PhD program a smooth experience. Special thanks to Debbie DeFord for personally receiving hundreds of mailed packages on my behalf, as well as to Cathy Thornton, Javonnia Thomas, Alison Morrow, Jamie Meehan, and Elisa Ahmanson for true professionalism in their work, which made my life so much easier.

I would like to thank Dr. Dan Robinson for his exceptional graduate course series in non-linear optimization; his lucid explanations and outstanding lecture materials will continue to serve me in years to come. I would like to thank Professor Corneliu Rablau of Kettering University for his excellent instruction and mentorship during my undergraduate career and for continued support and friendship throughout my graduate studies.

I would like to thank those outside my academic community who have supported me spiritually during this time. Special thanks to Pastors Craig Garriott and Stan Long of Faith Christian Fellowship church for their years of dedicated service to the Baltimore community. I am further grateful to Pastor Bill Nelson, Dr. Karl and Debbie Dortzbach, and Dr. Dwight and Maria Schwartz for their personal mentorship. I would also like to thank Mrs. Luebke, who prayed for me daily over a span of several years.

I am also very grateful for those of my family. I would like to especially thank my wife, Dr. Monisha Billings, for standing by me while enduring continuous late work

nights with my physical or otherwise effective absence. Her loving encouragement and acts of support strengthened my resolve to press on. I am also grateful for my daughter of six months, Hadassah, who encouraged me with smiles and cheer morning and evening. I am grateful to my parents, Dan and Susan Billings, for a strong spiritual foundation and for a childhood that explored my deepest dreams and encouraged the development of my interests and abilities. Special thanks to my father for his understanding of my aptitudes and for pointing me towards the field of engineering. I am thankful to my brothers and sisters, Sarah, Ezra, Isaac, Lydia, and Gideon, for their love and support. I am likewise grateful to my father- and mother-in-law, Emmanual and Dr. Hansa Jayakumar, and my brother-in-law, Stephen Jayakumar, for their love and acceptance.

Finally, I give thanks to God, my hope in life, for sustaining me in body, mind, and spirit throughout this journey and for His ever-present love that is at work in the world.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Registration is the process of computing a spatial alignment between different sets of data that have been measured in different coordinate systems. Registration is an important enabling technology for many applications in modern medical practice. Examples of clinical capabilities enabled by registration include fusion of different imaging modalities in order to improve diagnostic capabilities in radiology[1] and computer-assisted guidance for interventional procedures.[2] Apart from its applications in modern day medicine, registration is a ubiquitous problem held in common with other fields, such as robotics and computer vision.

The goal of registration is to compute a spatial *transformation* that maps each set of data into a common coordinate system. Registration methods differ depending on the type of transformation that is used. At a high level, registration methods may be classified as *rigid* or *non-rigid* (i.e., *deformable*) in nature. Rigid registra-

tion methods compute *rigid-body transformations* that amount to repositioning the transformed data in space via a global *rotation* and *translation* without deforming the overall shape of the data. In other words, relative distances between points in the same dataset are preserved. Deformable methods, on the other hand, not only reposition but also deform the transformed data in some way. Deformable methods vary broadly from highly constrained, global transformations having few parameters, such as the *similarity* or *affine transformations* (which extend the rigid-body transformation by simple scaling or a general linear mapping of the data, respectively) to loosely constrained, locally computed transformations having many parameters, such as deformation fields. Local transformation methods typically incorporate a regularization term within the registration cost function in order to enforce a smoothness constraint on the locally computed deformations.

Registration methods for clinical applications may be divided into two major classifications: *intensity based* and *feature based*. Intensity-based methods minimize a cost function defined on the pixels of overlap between two images being registered. Feature-based methods, on the other hand, register geometric information describing two shapes being registered.

Examples of similarity metrics for image-based registration include sum of square differences (SSD), sum of absolute differences (SAD), correlation coefficient (CC), and information theoretic metrics, such as mutual information (MI). The SSD and SAD metrics are suitable for registering two images belonging to a single modality; CC

is useful when the intensities of the images being registered are linearly related; and MI is applicable to multimodal registration when no simple relationship between the image intensities exists. Depending on the metric used, a wide range of optimization techniques may be suitable to optimize these image-based metrics, including gradient-based techniques, such as gradient descent, quasi-Newton methods, etc. and non-gradiant-based techniques such as Powell's method, genetic methods, etc.[3]

Feature-based registration methods are commonly used to register positional data, such as the positions of salient landmarks or point clouds and mesh models used to delineate the surface of an object. Historically, geometric distance criteria for computing the optimal registration solution are the most common (e.g., [4]). Beyond positional data, feature-based registration methods may also incorporate other types of shape descriptors, such as surface curvature, surface normals, color, etc. (e.g., [5–7]). Like the image-based metrics, various forms of optimization may be suitable to optimize these feature-based registration metrics.

One difficulty held in common by nearly all registration techniques is the problem of converging to an incorrect solution, called a *local optimum* or *local minimum*. This happens because directly solving the globally optimal solution is computationally intractable for most registration problems; thus, local optimization strategies must be employed. Therefore, it is required, in general, for a registration to begin close enough to the correct solution in order to successfully recover the correct data alignment.

For intensity-based methods, one approach that can be useful to minimize the

influence of local minima is to conduct the registration in a hierarchical approach by beginning the registration at low resolution to compute a rough global alignment first and then proceeding to progressively higher image resolutions in order to register the local details in each image. Another approach to combat local mimima, which can be useful for feature-based and some image-based metrics, is to begin the registration from many initial alignments and then choose the best registration outcome. For feature-based methods, computing the modes of the shape distribution can also provide a rough initial alignment. Non-iterative feature-based registration methods, such as spin-images,[8] have also been developed for computing a rough alignment given no prior information concerning the transformation. In the case of medical applications, domain knowledge concerning the data being registered and concerning how it was acquired is often enough to determine an initial estimate of the true alignment prior to conducting a registration.

Similar discussions and classifications as given in the preceding paragraphs concerning registration methods may be found in the following survey articles [1, 3, 9].

The work of this dissertation first began as an interest in the problem of registering images from intraoperative tracked B-mode ultrasound to preoperative computed tomography (CT) imaging. Being unsatisfied with the performance of intensity-based information theoretic registration methods for registering ultrasound and CT images, we were interested in the idea of using intraoperative range imaging to improve the accuracy of these registrations by combining a surface-based registration performed

on the range image data with an intensity-based registration performed on the ultrasound image data. We first explored this idea in [10], where the need for additional information beyond the intensity-based registration was established and the potential for using a time-of-flight camera in order to acquire the range data was explored. The problem we were then faced with was how to combine the respective cost functions for each type of registration within a common optimization scheme. One of our goals was to devise a principled way to accomplish this without, for instance, simply assigning arbitrary weights to the different cost functions, which would have lacked a foundational justification. It was determined that representing each type of data as a set of features would enable the combined data to be simultaneously registered within a common feature-based registration scheme. Since all features are not created equal, it was further determined that incorporating a probabilistic framework within the feature-based registration would enable the various characteristics of each type of feature to be aptly modeled and the registration accuracy to be thereby improved. For this approach, some data sources provide information that is already in feature-based form, such as range imaging and tracking the tip of a pointer, for example, whereas other data sources require additional processing to convert the information into a set of features to be registered, such as segmenting a surface from tomographic imaging to form a surface mesh, segmenting surface contours from ultrasound images, or computing salient feature matches between multiple frames in a video sequence and triangulating the corresponding 3D feature locations. Figure 1.1 illustrates this idea

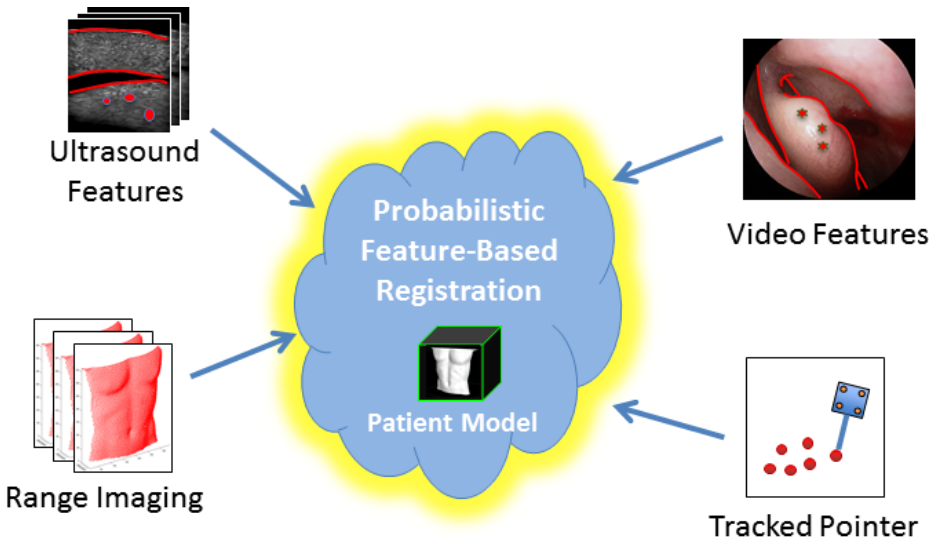

Figure 1.1: A graphical illustration of a probabilistic feature-based framework for registering various types of data.

of incorporating disparate types of feature-based data within a common probabilistic registration framework.

In support of the aforesaid vision, the goal of this work thus became to develop a new class of probabilistic feature-based registration algorithms that would enable combining different types of feature data and characterizing the uncertainty of each type of feature data within a coherent registration framework.

The use of feature-based methods for solving registration problems in clinical applications is well established, with the prior works being many and varied. For example, computer-assisted total hip replacement (THR) surgery may involve intra-operative sampling of points from an exposed region of a femur bone using a tracked pointer, which are then registered to a model of the bone surface segmented from preoperative CT imaging.[2] The use of range-imaging-based registration has been

demonstrated in various clinical settings, such as image-guided liver surgery,[11,12] cranio-maxillofacial surgery,[13] skull-base surgery,[14] neurosurgery,[15] and patient positioning for radiation therapy.[16] Surface-based techniques have also been proposed for registering intraoperative ultrasound to preoperative CT in various orthopedic applications, such as interventions of the spine[17] and pelvis.[18] Surface registration has also been applied to register multimodal tomographic images, such as CT and magnetic resonance imaging (MRI).[19–21] Other works have applied feature-based methods to register X-ray images to CT and CT angiography (CTA) by registering apparent bone contours[22] and blood vessels,[23] respectively. As a final example, feature-based structure from motion (SFM) methods have been demonstrated for registering endoscopic video to CT.[24]

1.1 Thesis Statement

Probabilistic algorithms for feature-based registration enable combining features of different types and uncertainty characteristics within a coherent optimization framework, bringing about improved registration performance.

1.2 Contributions

This dissertation presents contributions in several areas. The most notable area of contribution is the development of a new class of algorithms for feature-based reg-

Table 1.1: Summary comparison of the primary differences between ICP and the algorithms reported in this dissertation.

Alg.	**Feature Type**	**Noise Model**
ICP	3D positions	isotropic
IMLP	3D positions	anisotropic
IMLOP	3D positions & 3D orientations	isotropic
G-IMLOP	3D positions & 3D orientations	anisotropic
P-IMLOP	3D positions & 2D projected orientations	anisotropic
V-IMLOP	3D positions and 2D perspective-projected positions & orientations	anisotropic

istration, each of which are probabilistic variants on the Iterative Closest Point (ICP) algorithm.[4] A summary comparison of ICP and the various algorithms reported in this dissertation is shown in Table 1.1 and Figure 1.2, which respectively tabulates and graphically illustrates the differences in the probabilistic models associated with these algorithms. A generic strategy for incoporating deformable registration within the probabilistic registration frameworks of these algorithms is also described for shape deformations based on statistical shape models. Although applications in interventional medicine have been the focus for developing these algorithms, they are nonetheless broadly applicable to applications in related fields, such as robotics and computer vision. As a sub-contribution, a human leg phantom mimicking the clinical scenario of computer-assisted THR surgery was designed and constructed in order to assess the performance of various registration methods within this context. A final area of contribution is the design and development of an extensible software architecture that enabled rapid development of the various algorithms presented in this work. Following is a detailed outline of these contributions.

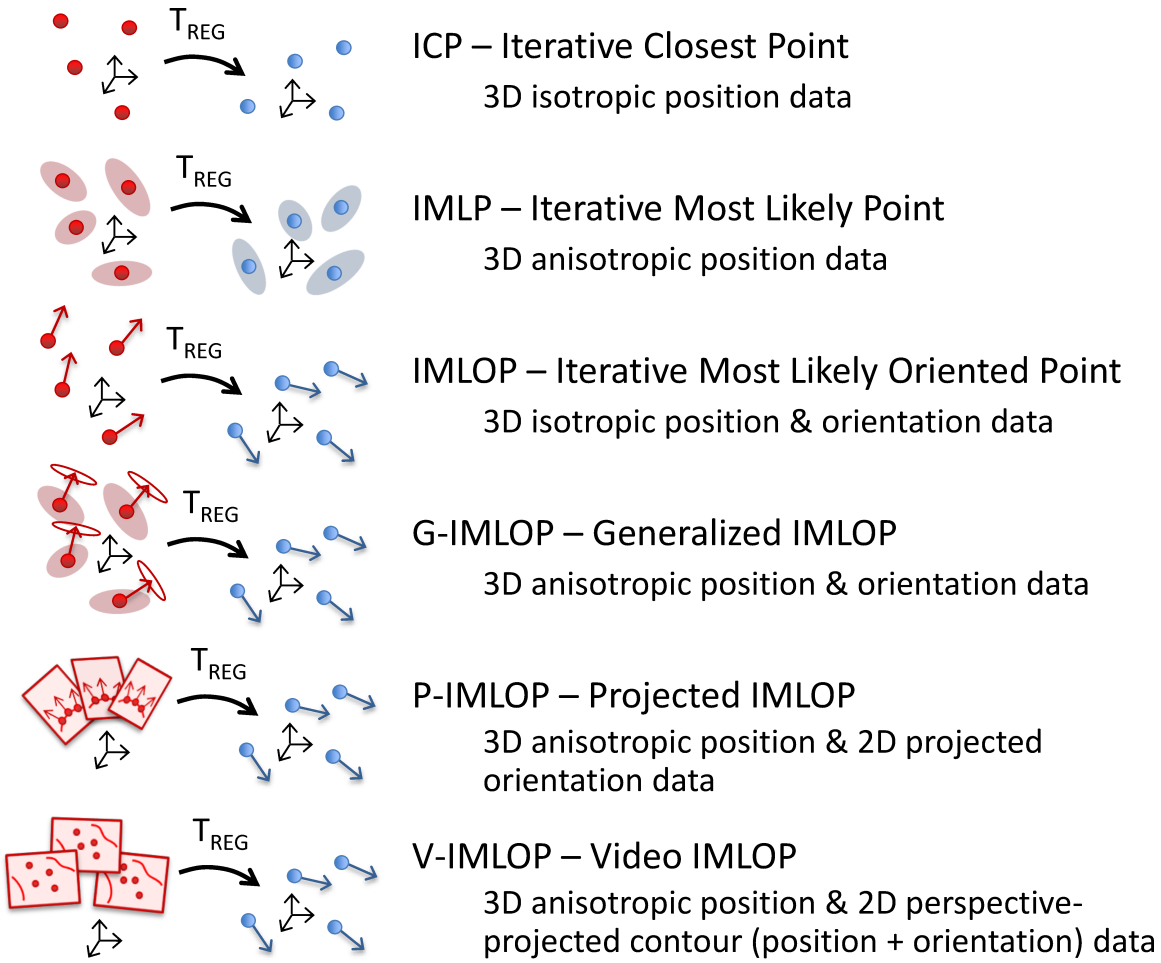

Figure 1.2: A graphical illustration of the primary differences between ICP and the algorithms reported in this dissertation.

Outline of Contributions

1. A systematic paradigm for designing and implementing a class of "most likely point" registration methods (Chapter 3)

2. Iterative Most Likely Point (IMLP) Algorithm[25] (Chapter 4)

   (a) a robust probabilistic algorithm for registering *positional* feature data that is characterized by *anisotropic* uncertainty; the algorithm dynamically adapts its noise model within each iteration in order to improve convergence towards the correct solution and in order to detect and mitigate outliers

   (b) an efficient implementation of the IMLP algorithm consisting of:

      i. *IMLP Correspondence Phase*: a fast PD-tree-based search for efficiently computing the most likely matches from a model shape under the assumption of anisotropic Gaussian uncertainty in the data- and/or model-shape positions

      ii. *IMLP Registration Phase*: an ad hoc solution to the generalized total least squares problem of registering corresponding point sets that are characterized by anisotropic uncertainty; this solution has the form of a modified Gauss-Newton approach that is more efficient, robust, and accurate than prior methods; further, this new solution is straightforward to implement using a linear least squares solver

3. Iterative Most Likely Oriented Point (IMLOP) Algorithm[26, 27] (Chapter 5)

   (a) a probabilistic algorithm for registering *positional* and *orientational* feature data (i.e., oriented points) that are characterized by *isotropic* uncertainty; the algorithm incorporates a dynamically adaptive noise model to aptly weight the relative confidence in the position versus orientation data within each iteration

   (b) an efficient implementation of the IMLOP algorithm consisting of:

      i. *IMLOP Correspondence Phase*: a fast PD-tree-based search for efficiently computing the most likely matches from a model shape considering both the position and orientation components of the match error

      ii. *IMLOP Registration Phase*: a closed-form solution to the problem of registering corresponding oriented point sets assuming isotropic uncertainty in the positions and orientations; this solution is a special case of a more general solution found in [28]

   (c) a mechanism for autonomously assessing a registration outcome in order to determine, with high confidence, whether a registration has succeeded or failed to compute an accurate shape alignment

4. Generalized IMLOP (G-IMLOP) Algorithm[27] (Chapter 6)

(a) a probabilistic algorithm for registering *positional* and *orientational* feature data (i.e., oriented points) that are characterized by *anisotropic* uncertainty in the positions and/or orientations

(b) an efficient implementation of the G-IMLOP algorithm consisting of:

i. *G-IMLOP Correspondence Phase*: a fast PD-tree-based search for efficiently computing the most likely matches from a model shape considering both the position and orientation components of the match error, as well as the anisotropic uncertainty in these features

ii. *G-IMLOP Registration Phase*: a gradient-based solution to the problem of registering corresponding sets of oriented points assuming anisotropic uncertainty in the positions and orientations; this solution is computed using the gradient equations of G-IMLOP's objective function, which is optimized using an off-the-shelf, nonlinear, BFGS quasi-Newton optimizer

(c) a mechanism for autonomously assessing a registration outcome in order to determine, with high confidence, whether a registration has succeeded or failed to compute an accurate shape alignment

5. Projected IMLOP (P-IMLOP) Algorithm[29] (Chapter 7)

(a) a probabilistic algorithm for registering features from *tracked B-mode ultrasound* imaging; this algorithm registers *position* features characterized by

anisotropic uncertainty and *projected orientation* features defined within arbitrarily oriented planes in 3D space

(b) an efficient implementation of the P-IMLOP algorithm consisting of:

i. *P-IMLOP Correspondence Phase*: a fast PD-tree-based search for efficiently computing the most likely matches from a model shape considering both the position and projected orientation components of the match error, as well as the anisotropic uncertainty in the position data

ii. *P-IMLOP Registration Phase*: a gradient-based solution for the problem of registering projection-oriented data-shape features with a corresponding set of oriented model-shape features assuming anisotropic uncertainty in the feature positions; this solution is formed from the gradient equations of P-IMLOP's objective function, which is optimized using an off-the-shelf, nonlinear, BFGS quasi-Newton optimizer

(c) design and construction of a human leg phantom for assessing registration accuracy in application to computer-assisted total hip replacement surgery

6. Video IMLOP (V-IMLOP) Algorithm (Chapter 8)

(a) a robust probabilistic algorithm for registering *video-image-based* features to a 3D surface model, while supporting anatomical constraints on the registration solution; this algorithm registers 3D *scaled position* features and 2D *perspective projected* contour features that consist of *position* and *ori-*

entation components measured from video images; *anisotropic* uncertainty is supported for the positional features

(b) an efficient implementation of the V-IMLOP algorithm consisting of:

i. V-IMLOP Correspondence Phase

A. *Estimating the Occluding Contours of the Model Shape under Perspective Projection*: a GPU-based software framework for computing the perspective-projected occluding contours of a model shape as viewed from the imaging planes of a given set of virtual camera poses

B. *Computing the Most Likely Matches on the Model Shape*: fast PD-tree-based search techniques for efficiently computing the most likely matches from the model shape for the SFM features and for efficiently computing the most likely matches from the estimated contours of the model shape for the video contour features, while accounting for anisotropic data uncertainties and while accounting for both the position and orientation components of the contour features

ii. V-IMLOP Registration Phase with Anatomical Constraints

A. *Computing the Optimal Unconstrained Similarity Transformation to Align the Matched Feature Sets*: a gradient-based solution for the problem of registering the SFM and contour features with the

corresponding features from the model shape while accounting for anisotropic data uncertainties and while accounting for both the position and orientation components of the contour features; the solution is formed by computing the gradient equations of the objective function, which is optimized using an off-the-shelf, nonlinear, quasi-Newton optimizer

B. *Enforcing Anatomical Constraints on the Registration Solution*: incorporation of anatomical constraints on the computed transformations in order to restrict the estimated camera positions to feasible regions of the anatomy

7. A general probabilistic-based approach for incorporating deformable shape transformations within the registration frameworks of the probabilistic algorithms developed for this dissertation (Chapter 9)

   (a) detailed development of a deformable registration approach for registering a deformable model shape that is characterized by a statistical shape model (SSM), where the shape deformations computed by the registration are driven by the modes of the SSM

   (b) a worked-out example of applying the general SSM-based deformable registration approach, illustrating a deformable version of the IMLP algorithm

8. An extensible software architecture for supporting rapid development of ICP-based registration algorithms (Chapter 10)

1.3 Outline

An outline of this dissertation is now described. To provide background for the reader, Chapter 2 provides a detailed description of the ICP algorithm followed by an overview of prior work regarding feature-based registration algorithms, placing special emphasis on algorithms that are based on the ICP method. The principal direction (PD) tree is also introduced as background material, which is a data structure that is used and modified extensively for implementing the algorithms reported in this dissertation.

Following the background material, Chapter 3 introduces the "most-likely-point" paradigm as a systematic algorithm design process that is followed while developing each algorithm of this dissertation and which constitutes the first contribution of this work. This paradigm is also illustrated in this chapter by applying the design process to derive the ICP algorithm; as a side-effect of this derivation, the ICP algorithm is demonstrated to implicitly assume identically distributed isotropic uncertainty in the registered data.

Chapters 4–8 each describe one algorithm reported in this dissertation. These algorithms are all probabilistic in nature and, unlike ICP, explicitly define the prob-

ability distributions assumed for the uncertainties of the features being registered. Each of these chapters begin by discussing additional background and prior work as needed. The probabilistic model assumed by each algorithm is defined next, followed by a high level overview of the algorithm and low-level implementation details of each algorithmic phase. Experimental evaluation of the algorithm is then presented at the end of each chapter, followed by concluding remarks and a recap of the pertinent contributions. All experiments for these chapters were conducted on an Intel® Core™ i5-4200U dual-core mobile processor.

Chapter 4 presents the Iterative Most Likely Point (IMLP) algorithm, which is an algorithm for registering *positional* feature data characterized by *anisotropic* uncertainty. IMLP additionally includes machinery for dynamically updating the noise model in each iteration in order to improve convergence towards the correct solution and in order to eliminate the influence of outliers.

Chapter 5 presents the Iterative Most Likely Oriented Point (IMLOP) algorithm, which is an algorithm for registering *positional* and *orientational* feature data characterized by *isotropic* uncertainty in the positions and orientations. IMLOP additionally includes machinery for dynamically updating the noise model in each iteration in order to aptly weight the influence of the position vs. orientation data. A mechanism is described whereby registration outcomes are automatically assessed by the algorithm in order to detect when a registration has failed to compute an accurate alignment between the shapes being registered.

Chapter 6 presents the Generalized Iterative Most Likely Oriented Point (G-IMLOP) algorithm, which extends IMLOP by registering *positional* and *orientational* feature data characterized by *anisotropic* uncertainty in the positions and/or orientations. A mechanism is described whereby registration outcomes are automatically assessed by the algorithm in order to detect when a registration has failed to retrieve an accurate alignment between the shapes being registered.

Chapter 7 presents the Projected Iterative Most Likely Oriented Point (P-IMLOP) algorithm, which is a special-purpose algorithm for registering features in tracked B-mode ultrasound data. This algorithm assumes *anisotropic* uncertainty in the *positional* features and for *orientational* features incorporates special machinery for *projecting surface normals* from a 3D model onto the ultrasound image planes in order to register the 2D normal orientations of surface contours that are measured in the ultrasound images. This algorithm is evaluated by a phantom registration study designed to mimic the clinical scenario encountered in computer-assisted total hip replacement surgery.

Chapter 8 presents the Video Iterative Most Likely Oriented Point (V-IMLOP) algorithm for registering *video-based* feature data to a 3D surface model. This algorithm incorporates registration of *scaled 3D point* measurements computed via SFM methods and *2D oriented-point* data representing contours segmented from the video images, which are registered to perspective projections of occluding contours from the 3D model. *Anisotropic* uncertainty is supported for both the 3D SFM features

and for the 2D contour features. Initial experiments for the V-IMLOP algorithm are presented using clinical patient data with both real and simulated video-based features for the registration.

Chapter 9 details a probabilistic-based approach for deformable registration that is applied as a direct extension of the algorithms developed for this dissertation in order to support deformable registration of model shapes that are characterized by statistical shape models (SSMs). An illustrative example of applying this extension is presented for the IMLP algorithm. A path forward is further described for incorporating other types of shape deformation.

Chapter 10 describes an extensible software architecture and design pattern that was developed to enable rapid development of the registration algorithms described in this dissertation.

A uniform convention for notation is used throughout this manuscript, which is defined in Appendix A. Other appendices also exist, which are introduced as they are needed within the text.

1.4 Published Work

Much of the material in this dissertation appears in the following published works:

1. S. D. Billings, E. M. Boctor, and R. H. Taylor, "Iterative most-likely point registration (IMLP): A robust algorithm for computing optimal shape align-

ment," *PLoS ONE*, vol. 10, no. 3, p. e0117688, 2015. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0117688

2. S. Billings and R. Taylor, "Iterative most likely oriented point registration," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2014*, ser. Lecture Notes in Computer Science, P. Golland, N. Hata, C. Barillot, J. Hornegger, and R. Howe, Eds. Springer International Publishing, 2014, vol. 8673, pp. 178–185. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10404-1_23

3. S. D. Billings and R. H. Taylor, "Generalized iterative most likely oriented-point (G-IMLOP) registration," *International Journal of Computer Assisted Radiology and Surgery*, vol. 10, no. 8, pp. 1213–1226, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11548-015-1221-2

4. S. D. Billings, H. J. Kang, A. Cheng, E. M. Boctor, P. Kazanzides, and R. H. Taylor, "Minimally invasive registration for computer-assisted orthopedic surgery: Combining tracked ultrasound and bone surface points via the P-IMLOP algorithm," *International Journal of Computer Assisted Radiology and Surgery*, vol. 10, no. 6, pp. 761–771, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11548-015-1188-z

5. S. Billings, A. Kapoor, M. Keil, B. J. Wood, and E. Boctor, "A hybrid surface/image-based approach to facilitate ultrasound/CT registration," in *SPIE, Medical*

Imaging 2011: Ultrasonic Imaging, Tomography, and Therapy, J. D'hooge and M. M. Doyley, Eds., vol. 7968, 2011, p. 79680V. [Online]. Available: `http://dx.doi.org/10.1117/12.878941`

Chapter 2

Background

2.1 Iterative Closest Point (ICP) Algorithm

A widely popular algorithm for feature-based registration is the ICP algorithm introduced by Besl and McKay.[4] The ICP algorithm is an iterative procedure for registering two shapes that are represented as sets of features. The algorithm operates by decomposing one of the shapes to be registered (the *data shape*) into a set of points (if not already in point form) and computing a spatial transformation to optimally align these points with the second shape (the *model shape*), which may be represented in various forms such as another point cloud or a mesh surface model. The algorithm consists of iterating two key sub-phases: a *correspondence phase*, wherein the closest

point on the model shape is computed for each data point, and a *registration phase*, wherein a spatial transformation is computed to optimally align the two sets of corresponding points. This process iterates until the two shapes converge upon a stable alignment.

From a high-level perspective, the algorithms developed in this dissertation are all variations on ICP's basic algorithmic procedure. Due to its importance in relation to this dissertation and due to its fundamental contributions to the field, the ICP algorithm is described in detail in this section in order to provide further background for the reader. Consider a data shape represented by a set of points $\mathbf{X} = \{\mathbf{x}_i\}$ and a model shape represented by Ψ (typically another point cloud or a mesh). The ICP algorithm seeks to compute the rigid-body transformation, $\mathbf{T}$, comprised of a rotation, $\mathbf{R}$, and translation, $\mathbf{t}$, that minimizes the sum of square distances between the two shapes

$$\mathbf{T} = \underset{[\mathbf{R},\mathbf{t}]}{\operatorname{argmin}} \sum_{i=1}^{n} \|\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|_2^2 \tag{2.1}$$

where $\mathbf{y}_i$ is defined as the point on the model shape, Ψ, that is closest to the transformed data point $\mathrm{T}(\mathbf{x}_i) = (\mathbf{R}\mathbf{x}_i + \mathbf{t})$. We define the *closest point correspondence operator* ($\mathrm{C_{CP}}$) as the operator that returns the point, $\mathbf{y}$, on some model shape, Ψ, that is closest by Euclidean distance to some data point, $\mathbf{x}$

$$\mathbf{y} = \mathrm{C_{CP}}(\mathbf{x}, \Psi) = \underset{\mathbf{y} \in \Psi}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{x}\|_2 \ . \tag{2.2}$$

Thus, $\mathbf{y}_i$ in Equation (2.1) is given by $\mathbf{y}_i = \mathrm{C}_{\mathrm{CP}}(\mathrm{T}(\mathbf{x}_i), \Psi)$. The ICP algorithm is formulated as a sequential iteration of two key steps:

1. *Correspondence Phase*: Compute the closest point, $\mathbf{y}_i$, on the model shape for each point, $\mathrm{T}(\mathbf{x}_i)$, of the transformed data shape (2.2).

2. *Registration Phase*: Compute the transformation, $\mathbf{T}$, that minimizes the sum of square distances between the matched point pairs from the correspondence phase (2.1).

The correspondence phase of ICP has an efficient implementation using a KD-tree search,[30, 31] and various closed-form solutions have been presented for the registration phase.[32–34]

As previously mentioned, the ICP algorithm and derivatives of it are susceptible to local minima due to the local nature of the optimization procedure. The algorithm must therefore be initialized close to the correct solution. How close is needed largely depends on the type of data being registered. For example, shapes having a large degree of asymmetry are more likely to tolerate wide initial misalignments compared to more symmetric shapes. Shapes represented by a dense set of features are also generally more likely to tolerate large misalignments and still register the shapes correctly than feature sets that represent a sparse sampling of the shapes being registered.

2.2 Prior Feature-Based Registration Algorithms

Besides the popular ICP algorithm, which was specifically introduced in the prior section, many other algorithms exist for feature-based registration. An early method that predates ICP is the "head-and-hat" algorithm by Pelizzari et al.,[19] which is a special purpose feature-based method for registering 3D images of the human head. Similar to ICP, the head-and-hat algorithm iterates between a correspondence and a registration phase. In the correspondence phase, a match for each point forming the data (i.e., hat) shape is chosen from the model (i.e., head) shape by finding the intersection with the model surface of the ray that extends from the data centroid through the data point being matched. In the registration phase, the registration transformation is updated in a derivative-free manner using Powell's method. Around the same time period as ICP, Champleboux et al.[35] presented a feature-based method for registering generic shape models, which approximates the closest-point distances between a model shape and the points comprising a data shape by using a hierarchical distance map, called the Octree-Spline, which is constructed around the model shape. The sum of square distances between the data shape and the model shape is then directly minimized (i.e., there is no separation of a correspondence and a registration phase) using the Levenberg-Marquardt method for nonlinear least-squares optimization; the required derivatives of the match distances are approximated based on the

derivatives of the distances computed from the Octree-Spline.

Following the introduction of ICP by Besl and McKay,[4] many variants on the standard ICP algorithm were proposed. Chen and Medioni[36] minimize point-to-plane square distances between the data-shape points and tangent planes positioned on the model surface at the closest-point match locations. They demonstrated the usefulness of this method for registering range images. Zhang[37] presented a robust ICP variant that incorporates robust statistics and adaptive thresholding to handle outliers and occlusions in the correspondence phase. Maurer et al.[38] introduced weighting terms in the registration phase for each point pair, which they use for outlier rejection and for normalization of non-uniform point densities. Others have sought to improve match selection by augmenting the match metric with additional information besides Euclidean distance. Sharp et al.,[5] for example, used feature invariants, such as curvature, to refine match selection. Armesto et al.[39] proposed an alternate metric-based distance function for the scan-matching problem in mobile robotics, which takes into account both translation and rotation error of the sensor. Their work was based on extending the 2D metric-based ICP (MbICP) method of Minguez et al.[40] to the 3D case. An interesting approach by Fitzgibbon[41] (LM-ICP) directly minimizes a model-data error function using the nonlinear Levenberg-Marquardt algorithm while providing robustness to the registration via a Huber kernel. Like Champleboux et al.,[35] their method optimizes the cost function directly without separating the optimization into a correspondence and registration phase, and their approach is made

efficient by pre-computing a distance transform on the target shape.

For registration of two point-cloud shapes, various authors have incorporated soft matching, where each point in the data shape is matched to every point in the model shape (rather than to just one point) with a varying weight or probability associated with each pairing. Early works pioneering this approach were presented by Gold et al.,[42] using the softassign technique for matching, and by Chui and Rangarajan[43] (TPS-RMP) and Granger and Pennec[44] (EM-ICP), using Gaussian mixture models (GMMs) optimized within an expectation maximization (EM) framework. In addition to rigid registration, Chui and Rangarajan also included a non-rigid method based on thin-plate splines. An alternate consistent and symmetric approach for non-rigid registration based on EM-ICP is given by Combs and Prima.[45] A modern variant of the EM-based methods, called Coherent Point Drift (CPD), was presented by Myronenko and Song,[46] including a closed-form M-step solution for the rigid-body alignment problem and using Gaussian radial basis functions for the non-rigid alignment problem. The CPD algorithm treats one point cloud as the centroids of a GMM, which is aligned by maximum likelihood to the second point cloud representing the data set. Robustness to outliers is enabled by additionally matching each point to the background using an outlier weighting parameter. An alternate approach presented by Tsin and Kanade[47] treats each set of points as kernel densities formed from Gaussian kernel functions centered at each point; the registration is computed by maximizing a kernel correlation (KC) metric between these two densities. Jian

and Vemuri[48] present a similar idea for rigid and non-rigid registration by forming GMMs from each point set and minimizing the L2 distance between the Gaussian mixtures. While algorithms incorporating soft matching tend to achieve higher accuracy and to have wider basins of convergence towards the global optimum, these algorithms also tend to be less efficient than algorithms that incorporate single-point matching, due to the exhaustive point pairings.

2.3 Principal Direction (PD) Tree

An important implementation concern for ICP-based algorithms is to employ an efficient search technique when computing the matches in the correspondence phase, as this is the primary computational bottleneck for ICP-based methods. Having an efficient search strategy is therefore critical to the usefulness of these algorithms in practice. As mentioned in the prior section, the standard technique for Euclidean distance (i.e., closest point) matching is to use a KD tree to efficiently locate the points of correspondence on the model shape. The algorithms reported in this dissertation all employ a variant of the KD tree called the *principal direction (PD) tree*[49] (also known as the PCA or covariance tree). Both the KD and PD trees are data structures for partitioning a geometric space into a hierarchy of nodes that are tractable for geometric-based searching. The primary difference between the KD and PD trees is that each node of the PD tree has a local coordinate system that is oriented based

on the geometric dispersion of the resident data rather then being axis-aligned with a global coordinate frame. For the search strategies employed by the algorithms reported in this dissertation, this difference enables the geometric bounds of nodes within the PD tree to be more compact, and thereby provide a potential boost in search efficiency.

The generic search strategy employed for implementing the correspondence phase of an algorithm reported in this dissertation uses a PD tree formed around the model shape. For each iteration of the correspondence phase, the PD tree is searched once for each point in the data shape, with each search returning the optimal point of correspondence on the model shape.

In order to construct the PD tree, it is assumed that the model shape is comprised of a set of geometric primitives, such as points for a point cloud model or triangles for a mesh model. This set of geometric primitives are abstractly referred to as the *datums* of the tree. For each datum, a reference position on the datum is identified by which the datum's position is represented within the tree. These *datum positions* are defined for the purpose of tree construction and are simply chosen to be any point that is located on the datum. For example, in the case of a mesh-based model shape, the triangle center points serve as good choices for the datum positions. For a point cloud model, the choice of datum position is trivial, since the datum itself is comprised of only a single point.

Construction of a PD tree begins by assigning all of the datums of the model shape

to a root node. Given a set of datums that comprise a node of the PD tree, the node is constructed as follows. First, the covariance of the datum positions is computed for the node, which takes the form of a covariance matrix. A local coordinate frame is then defined for the node, such that the coordinate axes align with the eigenvectors of the covariance matrix. A minimally sized oriented bounding box (OBB) is then defined in local node coordinates (whose sides are aligned with the local coordinate frame) that fully contains all of the datums belonging to the node. At this step, it may be necessary to expand the bounds of the bounding box beyond that of the datum positions in order to fully contain each datum. For example, in the case of a mesh model, the bounding box would be expanded to contain every vertex of each triangle within the node. For a given level of the PD tree, this step may result in some nodes having bounding boxes that slightly overlap; nonetheless, each datum will only belong to one node at each level in the tree. Both the transformation from global to local node coordinates and the bounds of the bounding box are stored as properties of each node. The final step in the node construction process is to recursively create two child nodes by partitioning the datums within the current (i.e., parent) node along the direction of greatest extent (i.e., along the eigenvector associated with the eigenvalue of greatest magnitude from the eigen decomposition of the covariance matrix of the datum positions). This process continues down each level of the tree until a minimal number of datums or a minimal node boundary size has been reached. Figure 2.1 graphically illustrates the layout of nodes forming the upper two levels of an example

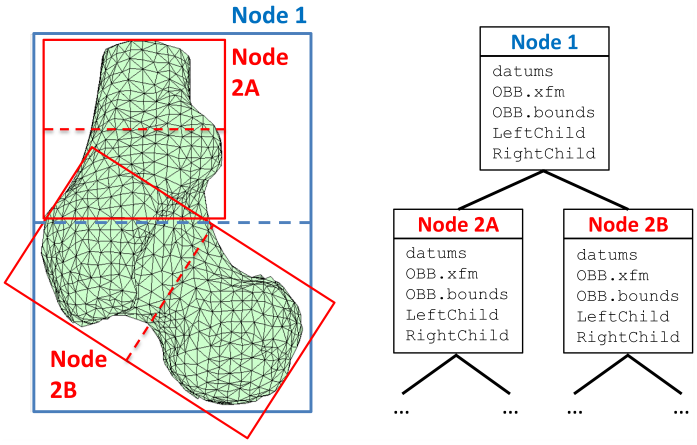

Figure 2.1: Illustration of nodes forming the upper two levels of a standard PD tree.

PD tree.

The purpose of constructing the PD tree is to enable efficient geometric-based searching of the model shape. Williams et al.[50] previously investigated using a PD tree for the problem of closest-point searching. To illustrate its use for closest-point searching, consider a data point, $\mathbf{x}$, for which we want to identify the closest point, $\mathbf{y}$, on the model shape. Consider as well that there is some starting candidate for the closest match, which we refer to as $\mathbf{y}_{\text{best}}$ and which lies at a distance d_{best} from the data point, $\mathbf{x}$. The search for the closest match begins from the root node of the PD tree and works downward to the leaf nodes. The problem that must be solved when first encountering each node in the PD tree is as follows: is it possible for a datum contained anywhere within the bounds of the node to lie at a distance less than d_{best} from $\mathbf{x}$? A simple solution to this problem is obtained by expanding the bounding box of the node by the distance d_{best} in all directions and then testing whether the data point, $\mathbf{x}$, lies within the bounds of the expanded node. If not, then it is not

possible for the node to contain a closer point than $\mathbf{y}_{\text{best}}$, and the node, along with all of its child nodes, may be disregarded in the continuing search; otherwise, the node is searched by propagating the search to its child nodes. When a leaf node is reached, the closest point on every datum within the leaf node is computed, and the candidate point, $\mathbf{y}_{\text{best}}$, and the match distance, d_{best}, are updated whenever a closer point is found. Once the recursive search that began at the root node is complete, then $\mathbf{y}_{\text{best}}$ will be the closest point on the model shape to $\mathbf{x}$.

As an optimization, since every datum must lie within the root node of the PD tree, a search may be started directly from the children of the root node rather than from the root node itself (provided that the child nodes exist), which reduces the number of node boundary tests for each search by one. For the registration algorithms described in this dissertation, a good starting candidate for $\mathbf{y}_{\text{best}}$ is the best match from the prior iteration. If no prior match is available (e.g., in the algorithm's very first iteration) then a good initialization may be obtained by traversing a straight path from the root node down to a leaf node of the PD tree; at each level of the tree, the next child node is chosen such that the data point, $\mathbf{x}$, lies on the same side of the node partition as the datums of the child node. When a leaf node is reached, then the reference position for any datum within the leaf node is used to initialize the search. Alternatively, any point on the model shape could be chosen to initialize the PD-tree search; however, this more naive approach may result in a far greater number of node searches during the first iteration.

Chapter 3

Most-Likely-Point Paradigm

At a high level, the algorithms presented in this dissertation all follow a common developmental approach, which is discussed here as the *most-likely-point paradigm*. Being variants on the ICP method, the proposed algorithms each have a correspondence phase and a registration phase, which are iterated until convergence. A primary difference compared to ICP is that the proposed algorithms explicitly incorporate probabilistic frameworks, which have at their foundation a probability density function (PDF) that describes the uncertainty in the features of the measured data shape, $\mathbf{X} = \{\mathbf{x}_i\}$, under an assumption of some correspondence with the model shape, Ψ. This PDF is unique to each algorithm and establishes the primary difference between the various algorithms.

If $\mathbf{y}$ is a model-shape point that is considered to be homologous (i.e., in correspondence) with a data-shape point, $\mathbf{x}$, then we can treat this correspondence as a

parameter of the expected distribution of $\mathbf{x}$ and denote the PDF of $\mathbf{x}$ as $f_{\text{match}}(\mathbf{x}\,;\mathbf{y})$. Given a definition for this PDF, solutions for the correspondence and registration phases of an algorithm can be determined.

In the correspondence phase, the *most likely match* is computed from the model shape for each point in the transformed data shape (rather than computing the closest match by Euclidean distance as used by ICP), while considering the transformation, $\mathbf{T}$, as known. Finding the most likely match for a given data point, $\mathbf{x}$, is accomplished by searching the model shape, Ψ, for the point, $\mathbf{y}$, that maximizes the probability of having generated $\mathbf{x}$. This computation is represented by the *most likely point correspondence operator* (C_{MLP})

$$\mathbf{y} = \text{C}_{\text{MLP}}(\mathbf{x}, \Psi) = \underset{\mathbf{y} \in \Psi}{\text{argmax}}\, f_{\text{match}}(\mathbf{x}\,;\mathbf{y}) \ . \tag{3.1}$$

Because the data point, $\mathbf{x}$, is known and the model correspondence point, $\mathbf{y}$, is an unknown parameter of the distribution that must be estimated, the PDF of $\mathbf{x}$ becomes a likelihood function over the set of all possible matches. Thus, we henceforth refer to $f_{\text{match}}(\mathbf{x}\,;\mathbf{y})$ as the *match likelihood function.*

In the registration phase, the problem of solving an optimal alignment between the data shape and the entire model is simplified to the problem of solving an optimal alignment between the data shape and the corresponding set of most likely point matches. A new transformation, $\mathbf{T}$, is computed that maximizes the probability of

the entire dataset, assuming a set of known matches, $\{\mathbf{y}_i\}$, which are given by the prior correspondence phase

$$\mathbf{T} = \underset{[\mathbf{R},\mathbf{t}]}{\operatorname{argmax}} \prod_{i=1}^{n} f_{\text{match}}(\mathrm{T}(\mathbf{x}_i)\,;\mathbf{y}_i) \ . \tag{3.2}$$

Since the data distributions are assumed to be independent, the *total match likelihood* that is maximized in (3.2) is obtained by multiplying the individual match likelihood functions for each data point.

The correspondence and registration phases are iterated until the algorithm converges. The author's implementations define convergence to be reached when the magnitudes of change in the rotation and translation that are computed by the registration phase fall below some threshold values for two consecutive iterations or when the number of iterations exceeds a maximum count. The change in translation is simply computed as the norm of the change in the translation vector. The change in rotation is computed by expressing the rotation increment in Rodrigues form[51] and taking the norm of the Rodrigues vector as the magnitude of angular change. The values used as thresholds are specified as inputs to the algorithm, being set by the user. Other criteria for convergence could alternatively be used as substitutes for these.

As previously mentioned for ICP-based methods, the primary computational bottleneck of this iterative algorithmic procedure occurs when computing the matches

in the correspondence phase. Thus, in practice, an efficient implementation for the correspondence phase is critical for algorithms based on this paradigm.

3.1 Most-Likely-Point Paradigm Illustrated with ICP

Although ICP does not explicitly define a match likelihood function, the most-likely-point paradigm can nonetheless be applied to derive the standard ICP algorithm. The ICP algorithm is derived here as an illustrative example of this paradigm.

Suppose that the points comprising the data shape are characterized by measurement noise that is independent identically distributed (iid), zero-mean, isotropic Gaussian with variance σ^2. Further assume that the points comprising the model shape are without noise (it can actually be assumed that the model points have iid, zero-mean, isotropic Gaussian noise as well, but for the simplicity of this derivation the model shape is assumed to be noise free). Assuming that a data point, $\mathbf{x}$, is homologous (i.e., in correspondence) with a given model point, $\mathbf{y}$, it follows that the match likelihood function that describes the probability of having generated the measurement, $\mathbf{x}$, from the location, $\mathbf{y}$, is defined as

$$f_{\text{match}}(\mathbf{x}\,;\mathbf{y},\sigma^2) = \frac{1}{(2\pi\sigma^2)^{3/2}} \cdot e^{-\frac{\|\mathbf{y}-\mathbf{x}\|_2^2}{2\sigma^2}} \tag{3.3}$$

where the correspondence, $\mathbf{y}$, is treated as the mean of the Gaussian distribution that generates the data point, $\mathbf{x}$.

Following the procedure prescribed for the correspondence phase, the task is to find the set of points, $\{\mathbf{y}_i\}$, on the model shape that maximize the match likelihood functions of the corresponding data points, $\{\mathbf{x}_i\}$. Maximizing the match likelihood function is equivalent to minimizing the negative log of the match likelihood function, which for ICP is

$$\mathrm{L}_{\mathrm{match,ICP}}(\mathbf{x}, \mathbf{y}, \sigma^2) = \frac{3}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{x}\|_2^2 \ . \tag{3.4}$$

Finding the point, $\mathbf{y}$, that minimizes (3.4) further simplifies to minimizing the Euclidean distance between the matched points

$$\mathrm{E}_{\mathrm{ICP}}(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{x}\|_2 \tag{3.5}$$

which is the *match error function* of ICP (2.2).

Following the procedure prescribed for the registration phase, the task is to compute the transformation, $\mathbf{T}$, that when applied to the data shape maximizes the total match likelihood as defined by (3.2). This is equivalent to minimizing the negative log of the total match likelihood, which for ICP is

$$\mathbf{T} = \underset{[\mathbf{R},\mathbf{t}]}{\operatorname{argmin}} \frac{3n}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}\|\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|_2^2 \ . \tag{3.6}$$

Minimizing (3.6) with respect to $\mathbf{R}$ and $\mathbf{t}$ further simplifies to minimizing the *total match error* function

$$\mathbf{T} = \underset{[\mathbf{R},\mathbf{t}]}{\operatorname{argmin}} \sum_{i=1}^{n} \|\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|_2^2 \tag{3.7}$$

which is the registration cost function of ICP (2.1).

This exercise of applying the most-likely-point paradigm to derive the ICP algorithm has demonstrated that ICP makes an implicit assumption of iid, zero-mean, isotropic, Gaussian noise for the registered data.

3.2 Contributions

The contributions from this chapter include:

1. A systematic paradigm for designing and implementing a class of "most likely point" registration methods

Chapter 4

Iterative Most Likely Point (IMLP) Algorithm

This chapter presents the Iterative Most Likely Point (IMLP) algorithm, which is an algorithm for registering positional shape data characterized by anisotropic uncertainty. Content of this chapter has appeared in [26].

The anisotropic noise model of the IMLP algorithm is motivated by anisotropic measurement uncertainties, which are prevalent in clinical data. For example, differences in intra-slice resolution versus inter-slice spacing give rise to anisotropic localization uncertainty for features in tomographic imaging. Ultrasound imaging is largely anisotropic, having the highest resolution axially and progressively lower resolution in the lateral and elevation directions.[52] Range imaging is also characterized by anisotropic measurement error; stereo-vision-based systems, for example, typically

have relatively high uncertainty in the depth direction.[53,54] The anisotropic uncertainty of stereo vision also applies to optical instrument tracking systems, such as the StealthStation® Surgical Navigation System (Medtronic, Minneapolis MN).

Other researchers have investigated probabilistic methods to improve upon the accuracy and flexibility of the ICP algorithm through incorporation of generalized noise models. In contrast to this, the ICP method and most variants implicitly assume an isotropic noise model, as was demonstrated in Section 3.1. Estépar et al.[55] introduce the robust Generalized Total-Least-Squares ICP (GTLS-ICP) algorithm for registration problems in medical imaging, which incorporates a generalized total-least-squares framework within the registration phase of the algorithm in order to account for anisotropic noise in the measured data points. Segal et al.[56] later employ a similar framework for their Generalized ICP (GICP) algorithm; instead of using the probabilistic framework to model measurement noise, however, they structure the noise model to approximately minimize a plane-to-plane square distance metric, which they demonstrate by range image registration to have an accuracy advantage compared to the point-to-plane method of Chen and Medioni.[36] However, the methods of Estépar et al. and Segal et al. use closest point matching, thereby following standard ICP procedure in the correspondence phase. Maier-Hein et al.[57] later introduced Anisotropic ICP (A-ICP), which primarily extends the works of Estépar et al. and Segal et al. by modifying the match criteria of the correspondence phase in order to minimize a Mahalanobis-distance metric defined by a set of noise-

model covariances. In lieu of an efficient method to compute such matches, A-ICP follows the procedure of first computing an initial registration using ICP and then, beginning from this initial registration, recomputing the registration with A-ICP while enforcing a user-defined bound on the search distance in the correspondence phase in order to reduce runtime. Finally, Moghari and Abolmaesumi[58] propose an ICP-like method based on the unscented Kalman filter algorithm, which is also able to account for anisotropic measurement error. Their method, which was further evaluated by Nazem et al. in [59], is an improvement over the extended Kalman filter algorithm of Pennec and Thirion.[60]

The probabilistic framework of the IMLP algorithm developed in this chapter is similar to the algorithmic frameworks of Estépar et al. (GTLS-ICP),[55] Segal et al. (GICP),[56] and Maier-Hein et al. (A-ICP).[57] Overall, our method is most similar to A-ICP and likewise incorporates a generalized noise model within both the registration and correspondence phases of the algorithm. A notable difference of our method is that point correspondences are computed to maximize a match likelihood function under an assumed multivariate Gaussian noise model, whereas A-ICP computes correspondences to minimize a Mahalanobis-based distance metric. As will be shown in this chapter, these approaches are not equivalent, and this difference in match criteria can significantly impact the accuracy of the computed registration. IMLP also includes distinct approaches for registering shapes of partial overlap and for handling outliers, which is based in part on dynamic updating of the noise model to account

for uncertainty in the matches. Other important differences include optimal and efficient implementations for the correspondence and registration phases of the IMLP algorithm; an overview of these important contributions is discussed in the following paragraphs along with further comparison to the prior work.

As an important implementation concern, we devise a novel search strategy in order to efficiently compute the most likely matches in the correspondence phase, which enables IMLP to run efficiently. Our strategy is based on a modified *principal direction (PD)-tree* search. A description of the standard PD-tree search technique for closest point (rather than most likely point) matching is found in Section 2.3. The efficient correspondence search technique is an essential element of the IMLP algorithm, because the primary computational bottleneck for ICP-based methods occurs during the correspondence search. In relation to prior work, as already mentioned the GTLS-ICP[55] and GICP[56] algorithms address the issue by simply using closest point matching, which has an efficient implementation based on the KD-tree data structure.[30] The A-ICP[57] algorithm, which cannot locate matches using a standard KD-tree search due to its anisotropic match criteria, addresses the efficiency concern by first registering the shapes with an alternative algorithm (i.e., ICP) and then performing a follow-up registration using A-ICP. This approach minimizes the number of iterations required for A-ICP to terminate. In addition, A-ICP imposes a distance bound on the search radius in order to limit the pool of match candidates for each sample point, with the pool of match candidates being found using a KD

tree. One drawback of this approach is that locating the best match, as defined by the match criteria, cannot be guaranteed. Further, the search within the pool of match candidates is performed exhaustively, which remains inefficient. The proposed correspondence search strategy for IMLP, on the other hand, is both efficient and guarantees that the most optimal match, as defined by the match criteria, is always selected from the model shape.

A second implementation concern for IMLP regards solving the optimization problem of the registration phase, which computes the rigid-body transformation that optimally aligns the points of the data shape with the set of corresponding model points computed from the prior correspondence phase, while taking into account the anisotropic noise model. While various closed-form solutions for minimizing the linear least squares square-distance metric of the standard ICP algorithm have been presented by Horn,[33] Arun et al.,[32] and Walker et al.,[34] the anisotropic probability framework incorporated by IMLP and by prior anisotropic methods leads to a nonlinear generalized total-least-squares (GTLS) optimization over the transformation parameters within the registration phase, for which no closed-form solution is known. Solving the GTLS problem thus requires more complex iterative methods of nonlinear optimization. The GTLS optimization developed for IMLP is based on a modified Gauss-Newton approach that has both speed and accuracy advantages compared to prior published solutions for this particular problem,[55,61] while also being straightforward to implement.

As alluded to above, the prior algorithms of Estépar et al. (GTLS-ICP),[55] Segal et al. (GICP),[56] and Maier-Hein et al. (A-ICP)[57] share in common with IMLP the same GTLS problem for computing an optimal alignment between corresponding point sets. Estépar et al. present an ad hoc solution that incorporates the iterative GTLS rotation estimation method of Ohta and Kanatani,[62] which is based on Kanatani's renormalization technique.[63] The Kanatani method solves the problem of computing rotation when translation is known using a quaternion parameterization of the rotation matrix. Estépar et al. extend this solution to solve the parallel problem of computing translation when rotation is known. Their approach for solving the full alignment problem is thus a dual-iterative one that first computes rotation assuming known translation and then computes translation assuming known rotation. This process iterates until both estimates converge. Another solution, which to our knowledge has not been applied within an ICP-based context, is presented in a paper by Matei and Meer[64] regarding their heteroscedastic errors-in-variables (HEIV) estimator. The HEIV estimator is a general-purpose method for solving a wide range of problems in computer vision through iterative solutions of a generalized eigenvector problem. The GTLS rigid-body point-set alignment problem is presented as an example application of the HEIV technique in [64]. Their solution is similar to Kanatani's renormalization approach[62] for solving only rotation in that both approaches involve solving eigenvalue problems and both use a quaternion parameterization for rotation. Rather than follow an ad hoc approach, Segal et al.[56] apply a generic conjugate-

gradient solver to optimize the GTLS cost function of GICP, in which rotation is parameterized as three Euler angles (determined by reference to their source code). Maier-Hein et al.[57] employ an ad hoc approach presented by Balachandran and Fitzpatrick in [61] and further analyzed in [65], which simultaneously solves for rotation and translation by successive linearization of the rotation matrix using a skew-matrix approximation for small rotation. One limitation of this method is that anisotropic noise is assumed for only one of the point sets, which may lead to inaccurate results when both point sets have anisotropic error distributions.

In contrast to the prior GTLS optimization methods, the modified Gauss-Newton-based approach developed for the IMLP algorithm supports anisotropic noise in both sets of points. As demonstrated in the experimental results of this chapter (Section 4.5.1), the proposed Gauss-Newton-based approach has several other advantages compared to the prior ad hoc methods of Estépar et al.[55] and Balachandran and Fitzpatrick,[61] including accuracy, speed, and stability. A further benefit of the proposed method, and of the prior ad hoc methods, is that only a linear least squares solver is required for its implementation; thus, the software dependency for a nonlinear optimization library is avoided.

4.1 Probabilistic Model

The probabilistic framework of IMLP incorporates a generalized Gaussian noise model that accounts for anisotropic errors in both the data- and model-shape positions. The errors on the measurements of these points are assumed to be independent, zero-mean, multivariate Gaussian distributed with unconstrained covariance. Thus, the match likelihood function of IMLP for a data point, $\mathbf{x}$, that is transformed by a current registration estimate, $[\mathbf{R}, \mathbf{t}]$, is defined as

$$f_{\text{match}}(\mathbf{x}\,;\mathbf{y}, \boldsymbol{\Sigma}_{\text{x}}, \boldsymbol{\Sigma}_{\text{y}}, \mathbf{R}, \mathbf{t}) = \frac{1}{\sqrt{(2\pi)^3 |\mathbf{R}\boldsymbol{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\text{y}}|}} \cdot e^{-\frac{1}{2}(\mathbf{y}-\mathbf{R}\mathbf{x}-\mathbf{t})^{\mathsf{T}}(\mathbf{R}\boldsymbol{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}}+\boldsymbol{\Sigma}_{\text{y}})^{-1}(\mathbf{y}-\mathbf{R}\mathbf{x}-\mathbf{t})} \quad (4.1)$$

where $\mathbf{y}$ is the model-shape point that is assumed to be in correspondence with the data-shape point, $\mathbf{x}$, and where $\boldsymbol{\Sigma}_{\text{x}}$ and $\boldsymbol{\Sigma}_{\text{y}}$ are covariances of uncertainty in the positions of $\mathbf{x}$ and $\mathbf{y}$, respectively. Note that the correspondence position, $\mathbf{y}$, serves as the mean of the distribution of $\mathbf{x}$.

In the correspondence phase of the IMLP algorithm, the model shape must be searched for the point, $\mathbf{y}$, with the associated noise model, $\boldsymbol{\Sigma}_{\text{y}}$, that maximizes the

match likelihood function of (4.1). This operation is equivalent to minimizing

$$\mathrm{E}_{\mathrm{IMLP}}(\mathbf{x}, \mathbf{y}, \mathbf{\Sigma}_{\mathrm{x}}, \mathbf{\Sigma}_{\mathrm{y}}, \mathbf{R}, \mathbf{t}\,) = \log|\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\mathrm{y}}| + (\mathbf{y} - \mathbf{R}\mathbf{x} - \mathbf{t})^{\mathsf{T}}(\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\mathrm{y}})^{-1}(\mathbf{y} - \mathbf{R}\mathbf{x} - \mathbf{t}) \quad (4.2)$$

which is the *match error function* for the IMLP algorithm.

In the registration phase of the IMLP algorithm, the transformation must be solved that maximizes the total match likelihood function (3.2). This operation simplifies to computing the transformation, $\mathbf{T}$, that minimizes the following *total match error function*

$$\mathbf{T} = \underset{[\mathbf{R},\mathbf{t}]}{\operatorname{argmin}} \sum_{i=1}^{n} \log|\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}i}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\mathrm{y}i}| + \sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t})^{\mathsf{T}}(\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}i}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\mathrm{y}i})^{-1}(\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}) \ . \quad (4.3)$$

In practice, the log term is dropped for the registration phase (discussed in Section 4.4), leading to the the following simplified registration cost function for the IMLP algorithm

$$\mathbf{T} = \underset{[\mathbf{R},\mathbf{t}]}{\operatorname{argmin}} \sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t})^{\mathsf{T}}(\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}i}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\mathrm{y}i})^{-1}(\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}) \ . \quad (4.4)$$

4.2 Algorithm Overview

In this section, a high-level overview of the IMLP algorithm is described, which is followed by later sections describing the low-level implementations of each algorithmic phase. Algorithm 4.1 provides a summary of the IMLP algorithm, which is referred to repeatedly in the discussion that follows.

The *measurement-error covariances* for the data point set and for a corresponding set of model points are represented by $\mathbf{\Sigma}_{\mathrm{X}} = \{\mathbf{\Sigma}_{\mathrm{x}i}\}$ and $\mathbf{\Sigma}_{\mathrm{Y}} = \{\mathbf{\Sigma}_{\mathrm{y}i}\}$, respectively. $\mathbf{\Sigma}_{\mathrm{Y}}$ is drawn from a larger set of covariances, $\mathbf{\Sigma}_{\Psi}$, that represents the entire model shape. $\mathbf{\Sigma}_{\Psi}$ may be either a superset of covariances or a rule for computing a covariance given some point on the model shape.

In addition to the covariances used to model the measurement error, IMLP includes explicit support for a second set of covariances, $\mathbf{\Sigma}_{\mathrm{SX}} = \{\mathbf{\Sigma}_{\mathrm{Sx}i}\}$ and $\mathbf{\Sigma}_{\mathrm{SY}} = \{\mathbf{\Sigma}_{\mathrm{Sy}i}\}$, which are useful for modeling the locally-linear surface regions surrounding each point of a point-cloud shape model. These *surface-model* covariances are added to the measurement-error covariances to obtain the complete noise model for each point. The motivation for including the surface-model covariances is to increase the variance in the surface-parallel directions in order to encourage match errors to distribute along the surface rather than perpendicular to the surface, thereby achieving closer alignment of the underlying surfaces being represented by the point-cloud shape models. This idea forms the basis of the GICP algorithm[56] and was also incorporated in the A-ICP algorithm.[57] The IMLP algorithm treats the surface-model

Algorithm 4.1. Iterative Most Likely Point (IMLP)

input : Data shape as point cloud: $\mathbf{X} = \{\mathbf{x}_i\}$
Model shape: Ψ
Measurement-error covariances: $\mathbf{\Sigma}_{\mathrm{X}} = \{\mathbf{\Sigma}_{\mathrm{x}i}\}, \mathbf{\Sigma}_{\Psi}$
Surface-model covariances: $\mathbf{\Sigma}_{\mathrm{SX}} = \{\mathbf{\Sigma}_{\mathrm{Sx}i}\}, \mathbf{\Sigma}_{\mathrm{S}\Psi}$
Upper bound on match uncertainty: $\sigma^2_{\max}$ (default: ∞)
Chi-square threshold value for outliers: χ^2_{thresh} (default: 7.81)
Outlier variance expansion factor: φ_{exp} (default: 9)
Initial transformation estimate: $[\mathbf{R}_0, \mathbf{t}_0]$

output: Final transformation $[\mathbf{R},\, \mathbf{t}]$ that aligns the shapes $\mathbf{X}$ and Ψ

1 Initialize transformation: $[\mathbf{R},\, \mathbf{t}] \leftarrow [\mathbf{R}_0,\, \mathbf{t}_0]$

2 Initialize noise model: $\sigma^2_{\mathrm{match}} \leftarrow 0$

3 Compute initial correspondences (Equ. 3.1):

$$[\mathbf{y}_i, \mathbf{\Sigma}_{\mathrm{y}i}, \mathbf{\Sigma}_{\mathrm{Sy}i}] \leftarrow \mathrm{C}_{\mathrm{MLP}}(\mathbf{x}_i, \Psi, \mathbf{I}, \mathbf{I}, \mathbf{R}, \mathbf{t}\,)$$

4 Skip to Step 6

5 Compute most likely correspondences (Equ. 3.1):

$$[\mathbf{y}_i, \mathbf{\Sigma}_{\mathrm{y}i}, \mathbf{\Sigma}_{\mathrm{Sy}i}] \leftarrow \mathrm{C}_{\mathrm{MLP}}(\mathbf{x}_i, \Psi, \mathbf{\Sigma}_{\mathrm{x}i} + \mathbf{\Sigma}_{\mathrm{Sx}i} + \sigma^2_{\mathrm{match}}\mathbf{I}, \mathbf{\Sigma}_{\Psi} + \mathbf{\Sigma}_{\mathrm{S}\Psi}, \mathbf{R}, \mathbf{t}\,)$$

6 Update the match-uncertainty noise-model term (Equ. 4.5):

$$\sigma^2_{\mathrm{match}} \leftarrow \min\left(\tfrac{1}{n_{\mathrm{inlier}}} \textstyle\sum_{i\,\in\,\mathrm{inliers}} \|\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|_2^2, \sigma^2_{\max}\right)$$

7 Identify outliers using a chi-square test (Equ. 4.7):

$(\mathbf{x}_i, \mathbf{y}_i)$ is outlier if $\mathrm{E}_{\mathrm{SqrMahalDist}}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{\Sigma}_{\mathrm{x}i}, \mathbf{\Sigma}_{\mathrm{y}i} + \sigma^2_{\mathrm{match}}\mathbf{I}, \mathbf{R}, \mathbf{t}\,) > \chi^2_{\mathrm{thresh}}$

and update the outlier noise-model terms (Equ. 4.8):

$$\varphi_i \leftarrow \begin{cases} \varphi_{\mathrm{exp}}\|\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|_2^2 & \text{if } (\mathbf{x}_i, \mathbf{y}_i) \text{ is an outlier} \\ 0 & \text{otherwise} \end{cases}$$

8 Set the noise-model covariances for the registration phase:

$$\mathbf{\Sigma}^*_{\mathrm{x}i} \leftarrow \mathbf{\Sigma}_{\mathrm{x}i} + \mathbf{\Sigma}_{\mathrm{Sx}i} + \tfrac{\varphi_i}{2}\mathbf{I}\,, \quad \mathbf{\Sigma}^*_{\mathrm{y}i} \leftarrow \mathbf{\Sigma}_{\mathrm{y}i} + \mathbf{\Sigma}_{\mathrm{Sy}i} + \tfrac{\varphi_i}{2}\mathbf{I} + \sigma^2_{\mathrm{match}}\mathbf{I}$$

9 Update the transformation to align the corresponding point sets (Equ. 4.4):

$$[\mathbf{R},\, \mathbf{t}] \leftarrow \underset{[\mathbf{R},\,\mathbf{t}]}{\operatorname{argmin}} \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t})^{\mathsf{T}} (\mathbf{R}\mathbf{\Sigma}^*_{\mathrm{x}i}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}^*_{\mathrm{y}i})^{-1} (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t})$$

10 **if** *not converged* **then** goto Step 5

covariances separately from the measurement-error covariances in order to exclude the surface model from the outlier detection stage, which was found to improve the algorithm's ability to reject outliers.

As seen in Algorithm 4.1, the noise models in IMLP's probabilistic framework include dynamically computed terms in addition to the measurement-error and surface-model covariances that are input by the user. The *match-uncertainty* term (σ^2_{match}) attempts to account for uncertainty in the match process by adding an isotropic variance to the noise models with a magnitude equal to the estimated amount of misalignment between the data and model shapes. In the initial iterations of the algorithm, the residual error between corresponding points is largely due to shape misalignment; thus, the input covariances may not accurately represent the underlying distribution of the match errors at first. As the algorithm iterates and the misalignment is reduced, the input covariances are expected to more accurately represent the distribution of the match errors. To account for this effect, we follow a similar approach to Estépar et al.[55] and model the match uncertainty as an isotropic noise term having a variance equal to the average square residual distance between the corresponding points. However, unlike Estépar et al., who include all match errors in the estimate, we only include match errors from the current set of inliers when computing the match-uncertainty term

$$\sigma^2_{\text{match}} = \frac{1}{n_{\text{inlier}}} \sum_{i \in \text{inlier}} \|\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|_2^2 \tag{4.5}$$

which has an intuitive appeal and which we found to improve IMLP's performance with respect to outlier rejection. Note that n_{inlier} represents the number of matches forming the current set of inliers. A detailed justification of this model for estimating match uncertainty is addressed by Sharp et al.[5]

Because the match-uncertainty term is isotropic, it may be added to the noise-model covariances of either the data or model points with the same effect. Since the match uncertainty intuitively affects the choice of correspondences, for the registration phase we choose to add this term to the covariances of the model points in Step 8 of Algorithm 4.1. However, for the correspondence phase in Step 5, the match-uncertainty term is added to the covariances of the data points, rather than the model points, because this choice reduces computational overhead. Note that, because computing σ^2_{match} requires having a set of correspondences in-hand, a fully isotropic noise model is used to initialize the correspondences in Step 3.

The match-uncertainty term described above also has importance for the chi-square outlier detection test in Step 7 of Algorithm 4.1. The match-uncertainty term enables the algorithm to converge robustly and quickly in the case of large initial misalignment by accounting for this misalignment in the noise model and preventing an overabundance of matches from being flagged as outliers based on the measurement-error covariances alone. In the case of registering shapes having only partial overlap, it could happen that the average square match distance remains large even at the properly registered alignment. In this case, it may be desirable to prevent the match-

uncertainty term from growing too large. To address this issue, a *match-uncertainty maximum threshold* ($\sigma^2_{\max}$) is defined as an optional input for the IMLP algorithm. If no value is specified by the user, then the maximum threshold is effectively disabled by setting it to a very large value.

Robustness to outliers is enabled via a chi-square test, which is used to identify outlier matches in Step 7 of Algorithm 4.1. Under an assumption of correspondence and of generalized Gaussian noise, the square Mahalanobis distance between matched points in 3D space is distributed as the sum of squares of three independent normalized Gaussian distributions, each representing a distribution along a different eigenvector of the noise-model covariance matrix. Thus, under the stated assumptions, the square Mahalanobis match distance has a chi-square distribution with three degrees of freedom.[66] Outliers are therefore detected by comparing each square Mahalanobis match distance

$$\mathrm{E}_{\mathrm{SqrMahalDist}}(\mathbf{x}, \mathbf{y}, \mathbf{\Sigma}_{\mathrm{x}}, \mathbf{\Sigma}_{\mathrm{y}}, \mathbf{R}, \mathbf{t}\,) = (\mathbf{y} - \mathbf{R}\mathbf{x} - \mathbf{t})^{\mathsf{T}}(\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\mathrm{y}})^{-1}(\mathbf{y} - \mathbf{R}\mathbf{x} - \mathbf{t}) \quad (4.6)$$

to the value of the inverse cumulative density function (CDF) of a chi-square distribution with three degrees of freedom evaluated at some probability, p. If a square Mahalanobis match distance exceeds this chi-square inverse CDF value (χ^2_{thresh}) then that match is considered an outlier. Thus, a matched point pair, $(\mathbf{x}, \mathbf{y})$, with corre-

sponding noise covariances, $\mathbf{\Sigma}_{\mathrm{x}}$ and $\mathbf{\Sigma}_{\mathrm{y}}$, is an outlier if

$$\mathrm{E}_{\mathrm{SqrMahalDist}}(\mathbf{x}, \mathbf{y}, \mathbf{\Sigma}_{\mathrm{x}}, \mathbf{\Sigma}_{\mathrm{y}}, \mathbf{R}, \mathbf{t}\,) > \mathrm{chi2inv}(p, 3) = \chi^2_{\mathrm{thresh}} \tag{4.7}$$

where $\mathrm{chi2inv}(p, 3)$ is the chi-square inverse CDF function with three degrees of freedom evaluated at probability p. χ^2_{thresh} is defined as an optional input parameter of the IMLP algorithm, which enables the user to adapt the algorithm to different percentages of outliers that may be present in the data. Setting this threshold to a very large value effectively disables outlier detection. Disabling outlier detection in this manner may be useful in cases where the data is known to be free from outliers or possibly cases where a large initial misalignment is present, although the match-uncertainty term ($\sigma^2_{\mathrm{match}}$) already functions as an automatic mechanism to account for large initial misalignment. When no chi-square inverse CDF threshold is specified by the user, the default threshold of 7.81 is used, which corresponds to a chi-square inverse CDF probability of $p = 0.95$.

To reduce the influence of outliers on the computed registration, a set of *outlier noise-model terms* ($\{\varphi_i\}$) are used to add further isotropic variance into the noise models of the matches identified to be outliers. The effect of this added variance is to reduce the influence of the outliers in the registration phase,[55] which occurs at Step 9 in Algorithm 4.1. If a match is determined to be an outlier then the outlier noise-model term corresponding to that match is set equal to the square Euclidean

distance between the matched points times the *outlier variance expansion factor*, φ_{exp}; otherwise, the outlier noise-model term is set to zero (4.8).

$$\varphi_i = \begin{cases} \varphi_{\text{exp}} \|\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\|_2^2 & \text{if } (\mathbf{x}_i, \mathbf{y}_i) \text{ is an outlier} \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

In the author's implementation, the outlier variance expansion factor, φ_{exp}, is set to 9 (which brings the outlier match errors within approximately 1/3 standard deviation relative to their noise models); this value may be reduced or increased to give respectively more or less weight to the outliers if desired.

Alternatively, in order to completely remove all outlier influence from the registration phase, any matches identified as outliers could be simply removed from the set of matches used to compute the registration in Step 9 of Algorithm 4.1. This strategy may be preferred for cases such as registering shapes having only partial overlap, since the systematic tug from the large body of non-overlapping points could then be significant enough to affect the final accuracy of the registration. For small to moderate percentages of random outliers, experience has shown that inflating the variance works just as well as disregarding the matches entirely.

Due to modifications made to the underlying noise models during each iteration, the IMLP algorithm cannot be guaranteed to converge. A similar scenario is encountered for many related ICP-based methods, and an in-depth discussion is provided by Zhang.[37] Because of the possibility for non-convergence, we have added cycling

detection as a further termination condition for IMLP. Cycling is detected by monitoring the value of the cost function being minimized within the registration phase. If the minimal cost computed by the registration phase increases twice within a period of four iterations and if the cost following the second increase is within a small tolerance of the cost following the first increase, then a cycle has been detected. In such cases, the algorithm terminates and returns the registration corresponding to the last iteration in which the cost function had decreased. This termination condition is primarily a precaution in order to ensure computational efficiency, since a cycling condition would terminate at the maximum iteration count in any case.

4.3 Correspondence Phase

This section describes an efficient implementation for the correspondence phase of the IMLP algorithm. Namely, an approach is described to efficiently compute the most likely match from the model shape, being the match that minimizes the match error function of (4.2), which is repeated below for ease-of-reference

$$\mathrm{E}_{\mathrm{IMLP}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\Sigma}_{\mathrm{x}}, \boldsymbol{\Sigma}_{\mathrm{y}}, \mathbf{R}, \mathbf{t}\,) = \log|\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\mathrm{y}}| + (\mathbf{y} - \mathbf{R}\mathbf{x} - \mathbf{t})^{\mathsf{T}}(\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\mathrm{y}})^{-1}(\mathbf{y} - \mathbf{R}\mathbf{x} - \mathbf{t}) \ .$$

It is important not to disregard the log term within (4.2) when computing the

most likely match, since the model covariance, $\mathbf{\Sigma}_{\mathrm{y}}$, may in general vary substantially across the model shape. Note that both the magnitude (i.e., the eigenvalues) and the orientation (i.e., the directionality of the eigenvectors) of $\mathbf{\Sigma}_{\mathrm{y}}$ may significantly change the value of the log term. Thus, even if the magnitude of noise is fixed at all points on the model shape, the log term may still have an impact for anisotropic distributions that have different orientations across the model shape. If both the magnitude and orientation of $\mathbf{\Sigma}_{\mathrm{y}}$ are constant across the entire model shape, then minimizing the match error function reduces to minimizing the square Mahalanobis distance term in (4.2).

Algorithms 4.2 and 4.3 summarize the efficient strategy described in this section for computing the most likely correspondences, with Algorithm 4.3 defining a node search subroutine that is called by Algorithm 4.2. Note that, in order to simplify the expressions within the algorithm summaries of this section, the covariance of a model-shape point is represented by a single covariance, $\mathbf{\Sigma}_{\mathrm{y}}$. However, as previously noted, the model-shape noise model for the IMLP algorithm is actually represented by two covariances, $\mathbf{\Sigma}_{\mathrm{y}}$ and $\mathbf{\Sigma}_{\mathrm{Sy}}$, in order to distinguish between the measurement-error and surface-model components of the total covariance. For the purposes of this section, no distinction between these components is required, and $\mathbf{\Sigma}_{\mathrm{y}}$ is considered to represent the total covariance of a model-shape point. In an actual implementation having both noise-model components defined over the model shape, each type of covariance would be stored and returned separately along with the most likely match.

Algorithm 4.2. PD-Tree Search for the Most Likely Correspondence

input : Data point: $\mathbf{x}$
Data-point noise model: $\mathbf{\Sigma}_{\mathrm{x}}$
PD tree containing the model shape (Ψ) and noise model ($\mathbf{\Sigma}_{\Psi}$): τ
Current transformation: $[\mathbf{R}, \mathbf{t}]$
Prior most likely match for this data point: $(\mathbf{y}_{\mathrm{prev}}, \mathbf{\Sigma}_{\mathrm{y\_prev}})$

output: Most likely match and its corresponding noise model: $(\mathbf{y}, \mathbf{\Sigma}_{\mathrm{y}})$

1 Initialize most likely match to the prior match:
$[\mathbf{y}, \mathbf{\Sigma}_{\mathrm{y}}, E_{\mathrm{best}}] \leftarrow [\mathbf{y}_{\mathrm{prev}}, \mathbf{\Sigma}_{\mathrm{y\_prev}}, \mathrm{E}_{\mathrm{IMLP}}(\mathbf{x}, \mathbf{y}_{\mathrm{prev}}, \mathbf{\Sigma}_{\mathrm{x}}, \mathbf{\Sigma}_{\mathrm{y\_prev}}, \mathbf{R}, \mathbf{t})]$

2 **if** τ.*Root has children* **then**

3 Search for more likely match in the left child of the PD-tree root node:
$[\mathbf{y}_{\mathrm{LChild}}, \mathbf{\Sigma}_{\mathrm{y\_LChild}}, E_{\mathrm{LChild}}] \leftarrow \mathrm{NodeSearch}(\tau.\mathit{Root}.\mathit{LChild}, E_{\mathrm{best}}, \mathbf{x}, \mathbf{\Sigma}_{\mathrm{x}}, \mathbf{R}, \mathbf{t})$

4 **if** $E_{\mathit{LChild}} < E_{\mathit{best}}$ **then** update most likely match:
$[\mathbf{y}, \mathbf{\Sigma}_{\mathrm{y}}, E_{\mathrm{best}}] \leftarrow [\mathbf{y}_{\mathrm{LChild}}, \mathbf{\Sigma}_{\mathrm{y\_LChild}}, E_{\mathrm{LChild}}]$

5 Search for more likely match in the right child of the PD-tree root node:
$[\mathbf{y}_{\mathrm{RChild}}, \mathbf{\Sigma}_{\mathrm{y\_RChild}}, E_{\mathrm{RChild}}] \leftarrow \mathrm{NodeSearch}(\tau.\mathit{Root}.\mathit{RChild}, E_{\mathrm{best}}, \mathbf{x}, \mathbf{\Sigma}_{\mathrm{x}}, \mathbf{R}, \mathbf{t})$

6 **if** $E_{\mathit{RChild}} < E_{\mathit{best}}$ **then** update most likely match:
$[\mathbf{y}, \mathbf{\Sigma}_{\mathrm{y}}, E_{\mathrm{best}}] \leftarrow [\mathbf{y}_{\mathrm{RChild}}, \mathbf{\Sigma}_{\mathrm{y\_RChild}}, E_{\mathrm{RChild}}]$

7 **else**

8 Search for more likely match in the PD-tree root node:
$[\mathbf{y}_{\mathrm{root}}, \mathbf{\Sigma}_{\mathrm{y\_root}}, E_{\mathrm{root}}] \leftarrow \mathrm{NodeSearch}(\tau.\mathit{Root}, E_{\mathrm{best}}, \mathbf{x}, \mathbf{\Sigma}_{\mathrm{x}}, \mathbf{R}, \mathbf{t})$

9 **if** $E_{\mathit{root}} < E_{\mathit{best}}$ **then** update most likely match:
$[\mathbf{y}, \mathbf{\Sigma}_{\mathrm{y}}, E_{\mathrm{best}}] \leftarrow [\mathbf{y}_{\mathrm{root}}, \mathbf{\Sigma}_{\mathrm{y\_root}}, E_{\mathrm{root}}]$

10 **end**

Algorithm 4.3. NodeSearch Function for the PD-Tree Search

input : PD-tree node being searched: $\mathcal{N}$
Data point: $\mathbf{x}$
Data-point noise model: $\mathbf{\Sigma}_{\mathrm{x}}$
Current transformation: $[\mathbf{R},\, \mathbf{t}]$
Current best match error: E_{best}

output: Position, noise model, and match error of the best match within the node: $[\mathbf{y}_{\mathrm{node}}, \mathbf{\Sigma}_{\mathrm{y\_node}}, E_{\mathrm{node}}]$

Initialize the best match within this node:
$[E_{\mathrm{node}},\, \mathbf{y}_{\mathrm{node}},\, \mathbf{\Sigma}_{\mathrm{y\_node}}] \leftarrow [E_{\mathrm{best}},\, 0,\, 0]$

Compute an ellipsoid bound ($\mathcal{E}$) centered at the transformed data point ($\mathbf{Rx} + \mathbf{t}$) within which candidates for a better match may be found:
$\mathcal{E} = \{\mathbf{z} \mid (\mathbf{z} - \mathbf{Rx} - \mathbf{t})^{\mathsf{T}}(\mathbf{\Sigma}_{\mathrm{sub}})^{-1}(\mathbf{z} - \mathbf{Rx} - \mathbf{t}) \leq E_{\mathrm{best}} - log_{\mathrm{min}}\}$
See Equations (4.12) and (4.13),(4.14), or (4.16)

if $\mathcal{E}$ *intersects* $\mathcal{N}.OBB$ **then**
  if $\mathcal{N}$ *is a leaf node* **then**
    foreach $datum_j \in \mathcal{N}$ **do**
      Compute the most likely match on $datum_j$ to get:
      $[\mathbf{y}_{\mathrm{datum}}, \mathbf{\Sigma}_{\mathrm{y\_datum}}, E_{\mathrm{datum}}]$ (Appendix B)
      if $E_{datum} < E_{node}$ **then** update the most likely match for this node:
      $[\mathbf{y}_{\mathrm{node}}, \mathbf{\Sigma}_{\mathrm{y\_node}}, E_{\mathrm{node}}] \leftarrow [\mathbf{y}_{\mathrm{datum}}, \mathbf{\Sigma}_{\mathrm{y\_datum}}, E_{\mathrm{datum}}]$
    end
  else
    Search left child node:
    $[\mathbf{y}_{\mathrm{LChild}}, \mathbf{\Sigma}_{\mathrm{y\_LChild}}, E_{\mathrm{LChild}}] \leftarrow \mathrm{NodeSearch}(\mathcal{N}.LChild, E_{\mathrm{node}}, \mathbf{x}, \mathbf{\Sigma}_{\mathrm{x}}, \mathbf{R}, \mathbf{t})$
    if $E_{LChild} < E_{node}$ **then** update the most likely match for this node:
    $[\mathbf{y}_{\mathrm{node}}, \mathbf{\Sigma}_{\mathrm{y\_node}}, E_{\mathrm{node}}] \leftarrow [\mathbf{y}_{\mathrm{LChild}}, \mathbf{\Sigma}_{\mathrm{y\_LChild}}, E_{\mathrm{LChild}}]$
    Search right child node:
    $[\mathbf{y}_{\mathrm{RChild}}, \mathbf{\Sigma}_{\mathrm{y\_RChild}}, E_{\mathrm{RChild}}] \leftarrow \mathrm{NodeSearch}(\mathcal{N}.RChild, E_{\mathrm{node}}, \mathbf{x}, \mathbf{\Sigma}_{\mathrm{x}}, \mathbf{R}, \mathbf{t})$
    if $E_{RChild} < E_{node}$ **then** update the most likely match for this node:
    $[\mathbf{y}_{\mathrm{node}}, \mathbf{\Sigma}_{\mathrm{y\_node}}, E_{\mathrm{node}}] \leftarrow [\mathbf{y}_{\mathrm{RChild}}, \mathbf{\Sigma}_{\mathrm{y\_RChild}}, E_{\mathrm{RChild}}]$
  end
end

Algorithm 4.3. NodeSearch Function for the PD-Tree Search (Continued)

Node Object Parameters:
Datums and corresponding noise-model covariances in this node:
$\{datum_i, \mathbf{\Sigma}_{yi}\}$
Oriented bounding box containing all datums in this node: OBB
Noise model used to form a lower bound on match errors within this node:
$\{\lambda_{\text{node\_min},i}\}$ and either $\mathbf{\Sigma}_{\text{node}}$ or $\lambda_{\text{node\_max}}$
(depends on the bounding method chosen in Step 2 of NodeSearch)

The search strategy for computing the most likely correspondences uses a PD tree that is formed around the model shape according to the standard PD-tree construction procedure described in Section 2.3, with a small modification being that additional parameters are stored in each node in order to support the anisotropic search. Figure 4.1 illustrates the difference in the node parameters for the IMLP algorithm compared to the standard PD tree. As described later in this section, multiple implementations of the IMLP node search are possible; Figure 4.1 illustrates the method used in the author's implementation, which is detailed in Sections 4.3.1 and 4.3.3. Due to the anisotropic nature of IMLP's match error function, the procedure described in Section 2.3 for searching the standard PD tree within the context of closest-point matching does not apply, and a modified search technique is therefore required.

In order to describe the anisotropic PD-tree search technique for IMLP, suppose that we are given a data-shape point, $\mathbf{x}$, having noise covariance, $\mathbf{\Sigma}_{\text{x}}$, and that we have a current candidate for the most likely match on the model shape, $\mathbf{y}_{\text{best}}$, that has

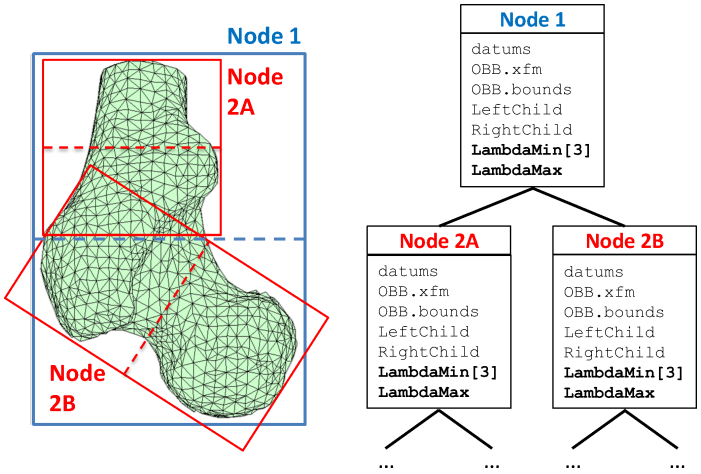

Figure 4.1: Illustration of nodes forming the upper two levels of a PD tree for the IMLP algorithm. Compared to the standard PD tree described in Section 2.3, the PD tree for IMLP includes additional node parameters to support anisotropic position-based searching; the topology of the PD tree is otherwise unchanged. The added node parameters are displayed in bold font in the figure, representing the PD-tree implementation described in Sections 4.3.1 and 4.3.3.

match error E_{best}. The search for the most likely correspondence point begins at the root node of the PD tree and progressively makes its way down the tree until reaching the leaf nodes. As described in Section 2.3, the problem that must be solved in order to search the PD tree efficiently is to derive an efficient technique for ruling out nodes during the PD tree search based on the oriented bounding boxes associated with the nodes, which define bounds on the positions of all datums belonging to each node. In other words, an efficient technique is required for answering the question: is it possible for a model point located anywhere within the node boundary to produce a match error that is lower than the match error, E_{best}, of the current match candidate, $\mathbf{y}_{\text{best}}$? To solve this problem, consider testing a match for which all inputs of the match error function (4.2) are known, except the position of the model point, $\mathbf{y}$. The goal of

determining whether any point located within the node's bounds can produce a match error less than E_{best} forms an inequality with respect to Equation (4.2). Introducing this inequality into (4.2) and shifting the log term to the opposite side defines the equation of an ellipsoid centered at the position of the transformed data-shape point, $(\mathbf{Rx} + \mathbf{t})$, as given by

$$(\mathbf{y} - \mathbf{Rx} - \mathbf{t})^{\mathsf{T}}(\mathbf{R}\boldsymbol{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\text{y}})^{-1}(\mathbf{y} - \mathbf{Rx} - \mathbf{t}) < E_{\text{best}} - \log|\mathbf{R}\boldsymbol{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\text{y}}| \ . \quad (4.9)$$

Any model point, $\mathbf{y}$, that produces a match error lower than E_{best} must be located within this ellipsoid boundary. Thus, the task now is to determine whether the ellipsoid so defined intersects the oriented bounding box of the node. If intersection exists, then it is possible that the node may contain a better match. If intersection does not exist, then the given node and all nodes below it cannot contain a better match, and and these nosed may be skipped in the continuing search. To compute the ellipsoid-OBB intersection test, we employ the efficient method described by Larsson.[67]

The problem of bounding the match error of a node is actually more complicated than indicated above, because different points on the model shape may have different noise-model covariances. Thus, the covariance, $\boldsymbol{\Sigma}_{\text{y}}$, may change for different datums within the same node. To address this issue, a substitute ($\boldsymbol{\Sigma}_{\text{sub}}$) for the match covariance ($\mathbf{R}\boldsymbol{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\text{y}}$) is required that produces a lower bound on the match error for any model point within the node compared to the match error that

is obtained when using the model point's true covariance. In other words, the ellipsoid bound that results from applying the substitute covariance, $\mathbf{\Sigma}_{\text{sub}}$, in (4.9) must fully contain all ellipsoid bounds that result from using any covariance in the set $\{(\mathbf{R}\mathbf{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\text{y}i})\}$ for all $\{\mathbf{\Sigma}_{\text{y}i}\}$ represented within the node. Note that the covariance expression $(\mathbf{R}\mathbf{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\text{y}})$ appears twice in (4.9), within both a log term and a square Mahalanobis-distance term. In the discussion that follows, we will consider independent replacements for the covariance expressions within each term.

4.3.1 Log-Component Bound

For the log term in (4.9) we seek a lower bound for the set of covariances represented within the node, since smaller log values increase the size of the ellipsoid boundary. Consider the two covariances, $\mathbf{R}\mathbf{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}}$ and $\mathbf{\Sigma}_{\text{y}}$, each having known eigenvalues, $\{\lambda_{\text{x},1}, \lambda_{\text{x},2}, \lambda_{\text{x},3}\}$ and $\{\lambda_{\text{y},1}, \lambda_{\text{y},2}, \lambda_{\text{y},3}\}$, respectively, which are arranged in order of increasing magnitude. A lower bound on the determinant of the sum of the two covariances is then given by

$$|\mathbf{R}\mathbf{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\text{y}i}| \geq \prod_{i=1}^{3}(\lambda_{\text{x},i} + \lambda_{\text{y},i}) \tag{4.10}$$

as proven by Fiedler.[68] It is clear from the eigen decompositions of the covariances

$$\mathbf{R}\mathbf{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\text{y}} = \mathbf{R}V_{\text{x}}\,\text{diag}(\lambda_{\text{x},1}, \lambda_{\text{x},2}, \lambda_{\text{x},3})V_{\text{x}}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}} + V_{\text{y}}\,\text{diag}(\lambda_{\text{y},1}, \lambda_{\text{y},2}, \lambda_{\text{y},3})V_{\text{y}}^{\mathsf{T}} \tag{4.11}$$

that this lower bound is achieved when the eigenvectors of $\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}}$ are in alignment with the eigenvectors of $\mathbf{\Sigma}_{\mathrm{y}}$ associated by eigenvalue rank, i.e., when $\mathbf{R}V_{\mathrm{x}} = V_{\mathrm{y}}$. Note that the expression $\mathrm{diag}(d_1, \ldots, d_n)$ represents a diagonal matrix formed from the listed elements, beginning from the upper left-hand corner. A lower bound on the log term for an entire node is therefore obtained by computing the smallest eigenvalue within each magnitude ordering among all covariance matrices, $\{\mathbf{\Sigma}_{\mathrm{y}i}\}$, represented in the node. The log term of (4.9) may then be replaced by

$$log_{\min} = \log \prod_{i=1}^{3} (\lambda_{\mathrm{x},i} + \lambda_{\mathrm{node\_min},i}) \tag{4.12}$$

where $\lambda_{\mathrm{x},i}$ are the eigenvalues of $\mathbf{\Sigma}_{\mathrm{x}}$ by magnitude order and $\lambda_{\mathrm{node\_min},i}$ are the smallest eigenvalues within each magnitude order among all covariances, $\{\mathbf{\Sigma}_{\mathrm{y}i}\}$, represented within the node. For example, $\lambda_{\mathrm{node\_min},2}$, which corresponds to the second largest magnitude order, is computed by selecting the smallest value from the set of all second largest eigenvalues (by absolute value) represented within the node.

Note that in order to implement the log bound, eigen decompositions for the data- and model-shape covariances need to be computed only once, since other noise-model components added by the IMLP algorithm are isotropic and thus uniformly increase each eigenvalue. As an optimization, nodes may use the same $\lambda_{\mathrm{node\_min},i}$ values as used by their parent node whenever these values remain within some factor of the values in use by the parent node. This enables the bound on the log term to be recomputed

only when doing so significantly affects the size of the ellipsoid boundary, rather than recomputing the log bound at every node visited.

4.3.2 Mahalanobis-Component Bound Method 1: Spherical Bound

In this and the two following two subsections, the problem of determining a substitute covariance for bounding the square Mahalanobis-distance term in (4.9) is addressed. For the square Mahalanobis-distance term, we seek a replacement for the match covariance $(\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}}+\boldsymbol{\Sigma}_{\mathrm{y}})$ that has a variance at least as large in any direction as that of any covariance from the set $\{(\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}}+\boldsymbol{\Sigma}_{\mathrm{y}i})\}$ for all $\{\boldsymbol{\Sigma}_{\mathrm{y}i}\}$ represented within the node, since increasing the variance in some direction increases the size of the ellipsoid bound in that direction.

The first method that we describe for bounding the square Mahalanobis-distance term provides the simplest and least compact bound. The idea is to replace the entire match covariance $(\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}}+\boldsymbol{\Sigma}_{\mathrm{y}})$ by the expression $(\lambda_{\mathrm{x\_max}}+\lambda_{\mathrm{node\_max}})\mathbf{I}$, where $\lambda_{\mathrm{x\_max}}$ is the largest eigenvalue of the data covariance, $\boldsymbol{\Sigma}_{\mathrm{x}}$, and $\lambda_{\mathrm{node\_max}}$ is the largest eigenvalue among all of the model-point covariances, $\{\boldsymbol{\Sigma}_{\mathrm{y}i}\}$, that are represented within the node. Performing this substitution along with the substitution of the log

bound simplifies (4.9) to

$$(\mathbf{y}-\mathbf{R}\mathbf{x}-\mathbf{t})^{\mathsf{T}}(\lambda_{\text{x\_max}}+\lambda_{\text{node\_max}})^{-1}\mathbf{I}(\mathbf{y}-\mathbf{R}\mathbf{x}-\mathbf{t}) < E_{\text{best}} - log_{\text{min}} \ . \qquad (4.13)$$

The advantage of this method is that the bounding ellipsoid simplifies to a bounding sphere (such as exists for the isotropic noise case), which results in a sphere-OBB intersection test with the node; this test is simpler and more efficient to compute than an ellipsoid-OBB intersection test. The simplicity of this method is offset, however, by the cost associated with forming a less compact bound, since a higher number of node searches may be performed as a result.

4.3.3 Mahalanobis-Component Bound Method 2: Simple Ellipsoidal Bound

An improvement over the first method for bounding the square Mahalanobis-distance term in (4.9) is achieved by finding a replacement for only $\mathbf{\Sigma}_{\text{y}}$ within the match covariance expression $(\mathbf{R}\mathbf{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}}+\mathbf{\Sigma}_{\text{y}})$. In this case, $\mathbf{\Sigma}_{\text{y}}$ is replaced by $\lambda_{\text{node\_max}}\mathbf{I}$, where $\lambda_{\text{node\_max}}$ is as defined in Section 4.3.2 for the first bounding method. Performing this substitution and substituting for the log bound simplifies (4.9) to

$$(\mathbf{y}-\mathbf{R}\mathbf{x}-\mathbf{t})^{\mathsf{T}}(\mathbf{R}\mathbf{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}}+\lambda_{\text{node\_max}}\mathbf{I})^{-1}(\mathbf{y}-\mathbf{R}\mathbf{x}-\mathbf{t}) < E_{\text{best}} - log_{\text{min}} \ . \qquad (4.14)$$

This method produces a bounding ellipsoid that is more compact than the bounding sphere of the prior method, yet remains simple to compute. As for the log bound, a node may re-use the $\lambda_{\text{node\_max}}$ value of a parent node whenever this value remains within some factor of the parent's value. This enables the ellipsoid bound to be recomputed only when doing so results in a significant reduction of the ellipsoid boundary rather than recomputing the covariance expression at every node visited.

4.3.4 Mahalanobis-Component Bound Method 3: Compact Ellipsoidal Bound

This final method of bounding the square Mahalanobis-distance term in (4.9) is the most complex but also the most compact bound. As previously mentioned, consider that a substitute for the match covariance must produce a bounding ellipsoid that fully contains all ellipsoid bounds that result from using any covariance within the set $\{(\mathbf{R}\boldsymbol{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\text{y}i})\}$ for all $\{\boldsymbol{\Sigma}_{\text{y}i}\}$ represented within the node. Further, consider that increasing the variance of $\boldsymbol{\Sigma}_{\text{y}}$ in any direction strictly increases the ellipsoid boundary defined by (4.9). The strategy then is to compute a new covariance that has a variance at least as large in all directions as any covariance, $\{\boldsymbol{\Sigma}_{\text{y}i}\}$, represented within the node, while producing a bounding ellipsoid that is as compact as possible. This is accomplished by computing a new covariance, $\boldsymbol{\Sigma}_{\text{node}}$, that represents the ellipsoid of minimal volume that fully contains the union of all ellipsoids produced by

each covariance, $\{\mathbf{\Sigma}_{yi}\}$, represented within the node.

$$\mathbf{\Sigma}_{\text{node}} = \underset{\mathbf{\Sigma}}{\text{argmin}} \, |\mathbf{\Sigma}^{-1}| \text{ such that the ellipsoid defined by } \{\mathbf{z} \in \mathbb{R}^3 \,|\, \mathbf{z}^{\mathsf{T}}\mathbf{\Sigma}^{-1}\mathbf{z} \leq 1\}$$
$$\text{fully contains the union of ellipsoids } \bigcup_{i \in \text{node}} \{\mathbf{z} \in \mathbb{R}^3 \,|\, \mathbf{z}^{\mathsf{T}}\mathbf{\Sigma}_{yi}^{-1}\mathbf{z} \leq 1\} \tag{4.15}$$

Note that the covariance, $\mathbf{\Sigma}$, computed in (4.15) is constrained to be a symmetric, positive-definite matrix. A method for approximating the minimal volume bounding ellipsoid of ellipsoids is addressed by Yildirim.[69] Also note that $\mathbf{\Sigma}_{\text{node}}$ is computed only once for each node when constructing the PD tree and is thereafter stored as a property of the node. Performing the substitutions for $\mathbf{\Sigma}_{\text{node}}$ and for the log bound modifies (4.9) to be

$$(\mathbf{y} - \mathbf{R}\mathbf{x} - \mathbf{t})^{\mathsf{T}}(\mathbf{R}\mathbf{\Sigma}_{\text{x}}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\text{node}})^{-1}(\mathbf{y} - \mathbf{R}\mathbf{x} - \mathbf{t}) < E_{\text{best}} - log_{\text{min}} \ . \tag{4.16}$$

As before, the covariance expression should be recomputed only when doing so substantially reduces the size of the ellipsoid boundary. Significant reduction of the ellipsoid bound may be determined by comparing the determinant of the node's $\mathbf{\Sigma}_{\text{node}}$ value to the determinant of the $\mathbf{\Sigma}_{\text{node}}$ value used by its parent node. If the node's determinant is within some small tolerance of the parent's determinant, then the node may continue to use the same match covariance as its parent. The optimal value for this tolerance could be found through experimentation by comparing runtimes using different values.

4.4 Registration Phase

This section describes an efficient and straightforward implementation for the registration phase of the IMLP algorithm. Namely, an approach is described to compute the transformation, $\mathbf{T}$, that minimizes the total match error function of (4.4).

Unlike in the correspondence phase, the covariance matrices, $\{\mathbf{\Sigma}_{yi}\}$, of the model points are now fixed. Although the value of the log term may still be affected by a change in rotation, this effect can be considered negligible relative to the impact of rotation on the square Mahalanobis-distance terms. Thus, the log term is disregarded in this phase, which simplifies the optimization considerably from Equation (4.3) to Equation (4.4), which is the minimization of a sum of square Mahalanobis distances and thus has the form of a nonlinear generalized total-least-squares (GTLS) problem. For ease-of-reference, Equation (4.4) is repeated below

$$\mathbf{T} = \underset{[\mathbf{R},\,\mathbf{t}]}{\operatorname{argmin}} \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t})^{\mathsf{T}} (\mathbf{R}\mathbf{\Sigma}_{xi}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{yi})^{-1} (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}) \ .$$

Due the nonlinear nature of (4.4), methods of solution require an iterative optimization approach. Before deriving the optimization approach, the problem is first expressed in an alternate form. It can be shown (Appendix C) that the unconstrained optimization of (4.4) (unconstrained subject to a valid rotation, that is) is equivalent

to the following constrained optimization

$$\mathbf{T} = \underset{[\mathbf{R},\,\mathbf{t}]}{\operatorname{argmin}} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}_i^*)^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{x}i}^{-1} (\mathbf{x}_i - \mathbf{x}_i^*) + \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{y}_i^*)^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{y}i}^{-1} (\mathbf{y}_i - \mathbf{y}_i^*)$$

$$\text{subject to:} \quad \mathbf{y}_i^* = \mathbf{R}\mathbf{x}_i^* + \mathbf{t} \tag{4.17}$$

where $\{\mathbf{x}_i^*\}$ and $\{\mathbf{y}_i^*\}$ represent the optimizer's estimates of the unknown, noise-free positions of the data-shape and model-shape point pairs, which, due to the correspondence assumption, are constrained to have perfect alignment. Thus, our goal is to solve the transformation parameters, $\mathbf{R}$ and $\mathbf{t}$, that minimize (4.17) subject to a perfect alignment constraint on the unknown, noise-free point positions.

Derivation of the following optimization strategy was particularly aided by the works of Gans[70] and Wentworth[71] regarding the topic of total-least-squares estimation. The first step in the derivation is to re-express the constraints of (4.17) as

$$\mathrm{F}_i(\mathbf{x}_i^*, \mathbf{y}_i^*, \mathbf{R}, \mathbf{t}) = \mathbf{y}_i^* - \mathbf{R}\mathbf{x}_i^* - \mathbf{t} = 0 \tag{4.18}$$

and then to linearize these constraints using a first-order Taylor-series expansion centered at the known values $\mathbf{y}_i$, $\mathbf{x}_i$, $\mathbf{R}_0$, and $\mathbf{t}_0$, where $\mathbf{R}_0$ and $\mathbf{t}_0$ are initial estimates of the transformation. Note that $\mathbf{R}_k$ and $\mathbf{t}_k$ are defined to be the estimates of the transformation parameters that are computed at each iteration, k. Performing a linearization of the rotation matrix leads to the skew approximation form for an

incremental rotation as defined by

$$\boldsymbol{\Delta R} \approx \mathbf{I} + \text{skew}(\boldsymbol{\Delta a}) \tag{4.19}$$

$$\text{skew}(\boldsymbol{\Delta a}) = \begin{bmatrix} 0 & -\Delta a_z & \Delta a_y \\ \Delta a_z & 0 & -\Delta a_x \\ -\Delta a_y & \Delta a_x & 0 \end{bmatrix} . \tag{4.20}$$

This parameterization enables representing small-angle rotations as a 3D vector, $\boldsymbol{\Delta a} = [\Delta a_x, \Delta a_y, \Delta a_z]^\mathsf{T}$. We note that using Lie algebra and exponential maps to parameterize the rotation, rather than the skew-approximation form described here, may also be a very effective approach for solving this problem. Note that skew($\mathbf{x}$)$\mathbf{y}$ is simply matrix notation for the cross product ($\mathbf{x} \times \mathbf{y}$); thus, the positions of $\mathbf{x}$ and $\mathbf{y}$ may be interchanged by negation, which is implicitly used in forming the Taylor-series expansion of the resulting constraint equations below. Using $\boldsymbol{\Delta a}$ to represent change in rotation and defining $\boldsymbol{\Delta t}$ to be change in translation, with some algebraic manipulation the constraints of (4.18) may be linearized to the approximate form

$$\begin{aligned} \mathrm{F}_i(\mathbf{x}_i^*, \mathbf{y}_i^*, \mathbf{R}, \mathbf{t}) &\approx \mathrm{F}_{\mathrm{L}i}^k(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\Delta a}, \boldsymbol{\Delta t}\,) \\ &= \mathrm{F}_i^0(\mathbf{x}_i, \mathbf{y}_i, \mathbf{R}_k, \mathbf{t}_k) - \mathbf{r}_{\mathrm{y}i} + \mathbf{R}_k \mathbf{r}_{\mathrm{x}i} + \text{skew}(\mathbf{R}_k \mathbf{x}_i)\boldsymbol{\Delta a} - \boldsymbol{\Delta t} = 0 \end{aligned} \tag{4.21}$$

which is linear with respect to change in rotation $\boldsymbol{\Delta a}$ and change in translation $\boldsymbol{\Delta t}$. Here we define $\mathrm{F}_i^0(\mathbf{x}_i, \mathbf{y}_i, \mathbf{R}_k, \mathbf{t}_k) = \mathbf{y}_i - \mathbf{R}_k\mathbf{x}_i - \mathbf{t}_k$, $\mathbf{r}_{\mathrm{y}i} = \mathbf{y}_i - \mathbf{y}_i^*$, $\mathbf{r}_{\mathrm{x}i} = \mathbf{x}_i - \mathbf{x}_i^*$, $\mathbf{R}_{k+1} = (\boldsymbol{\Delta R})\mathbf{R}_k \approx (\mathbf{I} + \text{skew}(\boldsymbol{\Delta a}))\mathbf{R}_k$, and $\mathbf{t}_{k+1} = \mathbf{t}_k + \boldsymbol{\Delta t}$, where k denotes each

iteration of the optimization procedure.

The next step in the derivation is to apply the method of Lagrange multipliers to enforce the linearized constraints while minimizing the cost function. The Lagrange function becomes

$$\mathcal{L}(\boldsymbol{\Delta a}, \boldsymbol{\Delta t}, \boldsymbol{\lambda}) = \sum_{i=1}^{n} \mathbf{r}_{\mathrm{x}i}^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{x}i}^{-1} \mathbf{r}_{\mathrm{x}i} + \sum_{i=1}^{n} \mathbf{r}_{\mathrm{y}i}^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{y}i}^{-1} \mathbf{r}_{\mathrm{y}i} + \sum_{i=1}^{n} \boldsymbol{\lambda}_i^{\mathsf{T}} \, \mathrm{F}_{\mathrm{L}i}^{k}(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\Delta a}, \boldsymbol{\Delta t}\,) \quad (4.22)$$

where $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_i\}$ represents the set of Lagrange multipliers with each $\boldsymbol{\lambda}_i$ being a 3-vector. The zero derivatives of the Lagrange function are now solved with respect to the residuals, $\{\mathbf{r}_{\mathrm{x}i}\}$ and $\{\mathbf{r}_{\mathrm{y}i}\}$, the change in transformation parameters, $\boldsymbol{\Delta a}$ and $\boldsymbol{\Delta t}$, and the Lagrange multipliers. After making substitutions between these differential equations, we finally obtain (4.23) for computing an incremental update of the transformation parameters, $\boldsymbol{\Delta p} = [\boldsymbol{\Delta a}, \boldsymbol{\Delta t}]^{\mathsf{T}}$.

$$\mathbf{J}^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \mathbf{J} \boldsymbol{\Delta p} = -\mathbf{J}^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} \mathbf{f}^0 \quad (4.23)$$

$$\boldsymbol{\Delta p} = \begin{bmatrix} \boldsymbol{\Delta a} \\ \boldsymbol{\Delta t} \end{bmatrix} \quad \mathbf{f}^0 = \begin{bmatrix} \mathbf{f}_1^0 \\ \vdots \\ \mathbf{f}_n^0 \end{bmatrix} \quad \mathbf{J} = \begin{bmatrix} \mathrm{skew}(\mathbf{R}_k \mathbf{x}_1) & -\mathbf{I} \\ \vdots & \vdots \\ \mathrm{skew}(\mathbf{R}_k \mathbf{x}_n) & -\mathbf{I} \end{bmatrix} \quad \boldsymbol{\Sigma} = \left[\mathbf{F}_x^0 \boldsymbol{\Sigma}_{\mathrm{x}} \mathbf{F}_x^{0\,\mathsf{T}} + \boldsymbol{\Sigma}_{\mathrm{y}} \right]$$

$$\mathbf{F}_x^0 = \begin{bmatrix} -\mathbf{R}_k & & \\ & \ddots & \\ & & -\mathbf{R}_k \end{bmatrix} \quad \boldsymbol{\Sigma}_{\mathrm{x}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathrm{x}1} & & \\ & \ddots & \\ & & \boldsymbol{\Sigma}_{\mathrm{x}n} \end{bmatrix} \quad \boldsymbol{\Sigma}_{\mathrm{y}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathrm{y}1} & & \\ & \ddots & \\ & & \boldsymbol{\Sigma}_{\mathrm{y}n} \end{bmatrix}$$

Here $\mathbf{f}^0$ is defined as a stacked vector of match residuals, $\mathbf{J}$ is the Jacobian matrix

of the constraints relative to the transformation parameters, and $\mathbf{\Sigma}$ is the complete covariance matrix for all matches. Since the match errors are assumed to be independent between matches, $\mathbf{\Sigma}$ has a 3×3 symmetric, positive-definite, block-diagonal structure for points in 3D. Simplifying this solution for registrations in 2D is trivial.

The update equation of (4.23) is a linear system of six equations having the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is 6×6 symmetric. Upon closer inspection, Equation (4.23) is recognized to be the normal equations of the linear system $\mathbf{\Sigma}_{\mathrm{x}i}^{-\frac{1}{2}}\mathbf{J}\mathbf{\Delta p} = -\mathbf{\Sigma}_{\mathrm{x}i}^{-\frac{1}{2}}\mathbf{f}^0$, where now $\mathbf{A} = \mathbf{\Sigma}_{\mathrm{x}i}^{-\frac{1}{2}}\mathbf{J}$ is $3n \times 6$. Equation (4.23) is further recognized to have the form of an update equation for the Gauss-Newton method of nonlinear least squares optimization. The solution to (4.17) is computed by iteratively solving (4.23) by linear least squares, with each solution providing an incremental update ($\mathbf{\Delta p}$) for the current transformation parameter estimates, $\mathbf{R}_k$ and $\mathbf{t}_k$, which are updated as

$$\mathbf{R}_{k+1} = \mathrm{R}(\mathbf{\Delta}\boldsymbol{a})\mathbf{R}_k\,, \quad \mathbf{t}_{k+1} = \mathbf{t}_k + \mathbf{\Delta t} \tag{4.24}$$

where $\mathrm{R}(\mathbf{\Delta}\boldsymbol{a})$ is as defined in the following paragraph. The linear system of (4.23) is re-solved following each update until the transformation estimates converge. This approach is nearly equivalent to the standard Gauss-Newton method, with a modification being that the covariance matrices are updated at each iteration.

In (4.24), $\mathrm{R}(\mathbf{\Delta}\boldsymbol{a})$ is defined to be a rotation matrix computed using the Rodrigues rotation formula,[51] which defines a rotation about an axis oriented along $\mathbf{\Delta}\boldsymbol{a}$ with

the rotation angle equal to $\|\boldsymbol{\Delta a}\|$ radians. Using the Rodrigues form, $\mathrm{R}(\boldsymbol{\Delta a})$, for updating the transformation parameters, rather than using the skew approximation form, $(\mathbf{I} + \mathrm{skew}(\boldsymbol{\Delta a}))$, ensures that $\mathbf{R}_{k+1}$ always satisfies the conditions for being a valid rotation matrix, namely that $\mathbf{R}\mathbf{R}^{\mathsf{T}} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$. Further discussion concerning the Rodrigues formula is found in Section 6.4.

Since the linear system of (4.23) is symmetric, an efficient and stable approach for solving the least-squares iterates is to use Cholesky or $\mathbf{L}\mathbf{D}\mathbf{L}^{\mathsf{T}}$ decomposition. The author's implementation employs the more general singular value decomposition (SVD), since the symmetric decompositions are not supported by the numerical libraries that were used. Because the linear system is small, SVD also provides reasonable efficiency. Note that, in the interest of efficiency, it is important to take advantage of the sparse structure of $\boldsymbol{\Sigma}$ when computing the matrix operations required to form this linear system.

The optimization terminates when the magnitude of incremental change in the transformation parameters fall below threshold values. In the author's implementation, convergence thresholds of 0.001 mm translation and 0.001 degrees rotation were used.

A summary of the approach described above for solving the GTLS problem of aligning two corresponding point sets characterized by anisotropic noise is provided below as Algorithm 4.4.

Algorithm 4.4. GTLS Registration of Corresponding Point Sets

input : Corresponding data and model point sets: $\mathbf{X} = \{\mathbf{x}_i\}$, $\mathbf{Y} = \{\mathbf{y}_i\}$
Noise-model covariances for the data and model points:
$\boldsymbol{\Sigma}_{\mathrm{X}} = \{\boldsymbol{\Sigma}_{\mathrm{x}i}\}$, $\boldsymbol{\Sigma}_{\mathrm{Y}} = \{\boldsymbol{\Sigma}_{\mathrm{y}i}\}$
Initial transformation estimate: $[\mathbf{R}_0, \mathbf{t}_0]$

output: Final transformation that aligns the corresponding point sets: $[\mathbf{R}, \mathbf{t}]$

1 Initialize the transformation: $[\mathbf{R}_k, \mathbf{t}_k] \leftarrow [\mathbf{R}_0, \mathbf{t}_0]$
2 Compute $\mathbf{f}^0$ using $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{R}_k$, and $\mathbf{t}_k$
3 Compute $\mathbf{J}$ using $\mathbf{R}_k$ and X
4 Solve incremental transformation $\boldsymbol{\Delta}\mathbf{p} = [\boldsymbol{\Delta}\boldsymbol{a}, \boldsymbol{\Delta}\mathbf{t}\,]^{\mathsf{T}}$ by linear least squares (4.23)
5 Update the transformation parameters: $[\mathbf{R}_k, \mathbf{t}_k] \leftarrow [\mathbf{R}(\boldsymbol{\Delta}\boldsymbol{a})\mathbf{R}_k, \mathbf{t}_k + \boldsymbol{\Delta}\mathbf{t}\,]$
6 Test for convergence:
if $\|\boldsymbol{\Delta}\boldsymbol{a}\| \leq \Delta\alpha_{\text{threshold}}$ *and* $\|\boldsymbol{\Delta}\mathbf{t}\| \leq \Delta t_{\text{threshold}}$ **then** Goto Step 2
7 Return the final transformation: $[\mathbf{R}, \mathbf{t}] \leftarrow [\mathbf{R}_k, \mathbf{t}_k]$

4.5 Experimental Results and Discussion

In this section, experimental registration studies are presented for the IMLP algorithm. We compare the IMLP algorithm to several competing methods under a wide range of test conditions including various isotropic and anisotropic noise levels, with and without outliers, and using different (i.e., mesh and point cloud) representations of various model shapes. Other methods evaluated for comparison with IMLP include ICP,[4] the robust variant of ICP by Zhang[37] (which we refer to as "Robust ICP"), GICP,[56] and CPD.[46] For the non-outlier cases, a close comparison is also made with GTLS-ICP[55] and A-ICP[57] using variants on our own method, IMLP-CP and IMLP-MD, respectively.

The two variants on IMLP directly compare the most-likely-point-matching criterion of IMLP with the closest-point-matching (CP) criterion used by GTLS-ICP

and the Mahalanobis-distance-matching (MD) criterion used by A-ICP. Since only the matching phase of IMLP-CP and IMLP-MD has been modified with respect to IMLP, this comparison directly evaluates the merits of the three criterion for computing matches: closest-point matching (GTLS-ICP, IMLP-CP), Mahalanobis-distance matching (A-ICP, IMLP-MD), and most-likely-point matching (IMLP).

GICP and CPD appear in the experiments involving a point-cloud model shape and not in the experiments involving a mesh-based model shape. This is because CPD is limited by design to point-cloud-to-point-cloud registration, and GICP as well is most suited to the context of registering non-continuous representations of two surfaces (i.e., point clouds).

For the GICP and CPD algorithms, we have used the implementations made publicly available by their respective authors. For the remaining algorithms (ICP, Robust ICP, IMLP, IMLP-CP, and IMLP-MD) our own implementations have been used. Minor changes were made to the source code of GICP and CPD in order to use the same termination criterion across all compared methods and, in the case of GICP, in order to orient the surface-model covariances directly along the known surface normal at each point rather than estimating the surface normals from neighboring points in the point cloud. These implementations are based on single-threaded programming in the C++ programming language. Thus, all methods were evaluated on level ground regarding the efficiency of the runtime environment, with an exception being that the CPD algorithm ran multi-threaded under certain settings (discussed later in

the results). As a further minor caveat, the CPD implementation uses Matlab as a front-end while incorporating a C-compiled mex library for the heavy lifting.

All registration methods were configured to terminate when the magnitude of change in the transformation parameters remained below threshold levels for two consecutive iterations or when a maximum iteration count was reached. The transformation thresholds were set to 0.001 mm translation and 0.001 degrees rotation with a maximum iteration count of 100 iterations (except where noted in the results). An advantage of using transformation thresholds as the basis for termination, rather than the cost function values, is that the need to normalize across the various cost functions employed by each algorithm is completely averted.

The algorithms implemented by the author (ICP, Robust ICP, IMLP, IMLP-CP, IMLP-MD) use the CISST[72] and WildMagic5[73] C++ libraries for numerical linear algebra and linear least squares computations. WildMagic5 is used for its efficient, non-iterative method of computing the eigen decomposition of a 3×3, symmetric, positive-definite matrix.

Before presenting a comparison of the algorithms described above, we begin the results section by evaluating the proposed approach for solving the GTLS problem of registering two corresponding point sets characterized by anisotropic noise (Algorithm 4.4). The proposed Gauss-Newton-based approach is compared to the prior methods of Estépar et al.[55] and Balachandran and Fitzpatrick,[61] which have also been proposed for solving this specific problem. However, one limitation of the method

as described by Balachandran and Fitzpatrick is that the anisotropic noise assumption is limited to the local coordinates of only one point set. All three methods share a commonality of being easy to program using a basic linear algebra library supporting a standard least squares solver.

4.5.1 Experiment 1: Generalized Total-Least-Squares Methods for Registering Corresponding Point Sets

In this study, we evaluate the proposed Gauss-Newton-based approach for computing the optimal rigid-body transformation that registers two corresponding point sets characterized by anisotropic measurement error, which was described in Algorithm 4.4. As previously stated, this problem has the form of a GTLS optimization problem and occurs in the registration phases of the IMLP algorithm and of closely related prior works.[55–57] The proposed GTLS method is evaluated on the basis of efficiency, accuracy, and stability relative to the prior methods for solving this problem proposed by Estépar et al.[55] and Balachandran and Fitzpatrick.[61] These anisotropic registration results are also compared to the isotropic registration result, which has a closed-form, least squares solution[32] and which constitutes the registration phase of the standard ICP algorithm.

Each method was evaluated using a Matlab-based implementation. For the method

of Balachandran and Fitzpatrick, the Matlab source code included in their paper was used.[61] For the other methods, Matlab implementations were programmed by the author, including an implementation of the rotation estimation method of Ohta and Kanatani,[62] which is a sub-component of the method by Estépar et al.

A high degree of instability was initially encountered when using the method of Estépar et al. under large translational offsets. A small modification sufficed to fix the issue, which involved applying the translation estimation prior to the first estimation for rotation. This modification was used throughout this study.

As previously noted, one limitation of the method by Balachandran and Fitzpatrick is that their method employs a single noise covariance that remains fixed as the algorithm iterates, due to the assumption of anisotropic noise in only one point set. Although noise in both point sets may be initially considered by combining the noise models to form a single covariance prior to calling their method (as in the covariance expressions of (4.2), for example), in this case the accuracy of the method still diminishes relative to the magnitude of rotational misalignment because the effective noise covariance is not updated as the method iterates.

Because of this limitation, a two-part study is conducted. The first study (Experiment 1A) investigates anisotropic noise present in both the data-shape and model-shape point sets. The second study (Experiment 1B) investigates anisotropic noise present in only the model-shape point set with isotropic noise present in the data-shape point set. For the second study, the assumption of a fixed effective covariance

becomes correct, since a change in the orientation of the data points has no impact on the effective noise covariance. We have included the method of Balachandran and Fitzpatrick in the evaluation of both studies, while computing an effective noise covariance as described in the foregoing paragraph.

The method of Balachandran and Fitzpatrick specifies initializing the anisotropic registration with the isotropic-noise solution before optimizing with respect to the GTLS cost function. We have performed a portion of the experiments both with and without isotropic initialization applied prior to each GTLS method. In order to better investigate the merit of the numerical machinery behind each GTLS approach, isotropic initialization was not used in Experiment 1A. In order to investigate the impact of initialization on each GTLS method, experiments were conducted both with and without isotropic initialization in Experiment 1B.

In order to compare all methods on level ground, several concerns had to be addressed. The first concern regards the termination criteria used by each method, which was implemented (or modified) to be when the magnitude of change in the estimated transformation parameters falls below 0.0001 mm and 0.0001 degrees or when the number of iterations exceeds 60. For the method of Estépar et al., a maximum iteration threshold of 20 was applied to the inner loop (i.e., to the rotation estimation component employing the method of Ohta and Kanatani) while the full outer loop was assigned the same maximum iteration threshold as the other GTLS methods. In practice, we found that these maximum iteration thresholds were only

reached under the condition of instability; thus, the iteration threshold was also used to automatically detect and count the occurrence of instability for each method.

The next concern regards the form of input afforded to each method. Every GTLS method compared requires some form of decomposition to be performed on the covariance matrices that define the anisotropic noise model, and these decompositions differ between the methods. To provide fair treatment, the noise covariances themselves are supplied as base-line input for each GTLS method. Since the implementation by Balachandran and Fitzpatrick was programmed to use pre-computed decompositions (i.e., the inverse square root) of the covariance matrices as input, we have added the required calculation to their method and changed the input to use the covariance matrices directly.

Another concern affecting runtime performance regards the style of Matlab coding. In order to obtain the best possible performance from each method, all matrix operations were fully vectorized in Matlab code, with the only exception being that a loop over the number of point pairs was required in order to compute the inverse square root of the covariance matrices for the method of Balachandran and Fitzpatrick, as no solution was identified to fully vectorize this operation across all point pairs. We have normalized for the runtime impact of this loop in Experiment 1B, which compares the method of Balachandran and Fitzpatrick on its own turf (i.e., with anisotropic noise in only one point set), by using a loop to compute the covariance decompositions required by the other GTLS methods as well. This loop-normalization was not per-

formed for Experiment 1A, however, as the runtime comparison with Balachandran and Fitzpatrick is already largely incongruent for that study due to their assumption of a fixed covariance (i.e., anisotropic noise in only one point set), whereas the other GTLS methods re-compute the covariance decompositions in every iteration. Another reason that fully vectorized implementations are used in Experiment 1A is in order to assess the full potential of the other methods.

As a final leveling of the playing field, a runtime normalization was applied in Experiment 1B for the assumption of a fixed covariance (i.e., anisotropic noise in only one point set). This was accomplished by creating variants of the implementations of the proposed Gauss-Newton-based method and of the method by Estépar et al. that assume, like the method of Balachandran and Fitzpatrick, that the effective noise covariance remains fixed for any orientation of the data point set. This test therefore provides a reasonable relative comparison of the runtimes that can be expected from each of the various GTLS optimization schemes.

The studies in Experiment 1 were conducted by first generating two noisy point sets with known correspondence and known ground-truth alignment, second applying a random misalignment between the point sets, and third registering the point sets using each registration method. To form a pair of corresponding point sets, a set of 50 points uniformly distributed within the interval [-100, 100] mm along each dimension in 3D space was randomly generated. These points served as the ground-truth points and also provided the ground-truth alignment of the two point sets. From this single

set of ground-truth points, two different noisy point sets were generated by adding zero-mean, multivariate, Gaussian noise, while using a different covariance for each point set. The two points generated from each ground-truth point were assigned as correspondences between the two point sets. The covariances were generated at random by forming a diagonal matrix of eigenvalues and multiplying on either side by a random rotation and its transpose

$$\text{Random Covariance} = \mathbf{R}\,\text{diag}(\lambda_1, \lambda_2, \lambda_3)\mathbf{R}^{\mathsf{T}} \ . \tag{4.25}$$

In Experiment 1A, involving anisotropic noise in both point sets, the eigenvalues of the noise covariances were set equal to $[0.5, 0.5, 2]\,\text{mm}^2$, with different random rotations being used for each set of points. In Experiment 1B, involving anisotropic noise in only one (the model) point set, these same eigenvalues were used for noising the model-shape points, whereas isotropic noise was generated for the data-shape points by setting all eigenvalues equal to $0.25\,\text{mm}^2$.

For each study, the randomized trials were divided into several bins according to the magnitude of initial misalignment in translation and rotation. For each bin, 1000 randomized trials were performed, and the results were averaged. For every trial, different sets of points, noise models, and misalignments were randomly generated and identically applied to each registration method. Registration accuracy was evaluated by computing the average distance between the un-noised point correspon-

Table 4.1: Experiment 1A: corresponding point set registration results with anisotropic noise present in both sets of points.

Trans	[10, 20]					[90, 100]				
Rot	**Alg**	**Iter**	**Time**	**RE**	**Inst**	**Alg**	**Iter**	**Time**	**RE**	**Inst**
[0, 15]	Isotropic	1.0	0.0001	0.439	0	Isotropic	1.0	0.0001	0.442	0
	Estépar	15.1	0.0106	0.423	0	Estépar	15.1	0.0106	0.424	0
	Balach.	28.1	0.0060	0.423	0	Balach.	32.9	0.0068	0.424	0
	Proposed	3.8	0.0014	0.422	0	Proposed	3.8	0.0013	0.423	0
[15, 45]	Isotropic	1.0	0.0001	0.443	0	Isotropic	1.0	0.0001	0.442	0
	Estépar	17.2	0.0120	0.432	0	Estépar	17.2	0.0120	0.431	0
	Balach.	34.1	0.0070	0.428	0	Balach.	37.5	0.0076	0.427	0
	Proposed	4.4	0.0015	0.424	0	Proposed	4.4	0.0015	0.423	0
[45, 90]	Isotropic	1.0	0.0001	0.442	0	Isotropic	1.0	0.0001	0.435	0
	Estépar	18.8	0.0131	0.456	0	Estépar	18.8	0.0130	0.450	0
	Balach.	38.1	0.0078	0.442	0	Balach.	40.3	0.0081	0.436	0
	Proposed	5.1	0.0017	0.424	0	Proposed	5.1	0.0017	0.416	0
[90, 150]	Isotropic	1.0	0.0001	0.446	0	Isotropic	1.0	0.0001	0.439	0
	Estépar	22.9	0.0154	0.469	2	Estépar	23.3	0.0157	0.466	2
	Balach.	39.4	0.0080	0.448	0	Balach.	41.9	0.0084	0.444	0
	Proposed	6.3	0.0021	0.430	0	Proposed	6.3	0.0021	0.421	0
[150, 180]	Isotropic	1.0	0.0001	0.444	0	Isotropic	1.0	0.0001	0.442	0
	Estépar	28.0	0.0184	0.475	60	Estépar	27.6	0.0181	0.477	59
	Balach.	42.0	0.0085	0.439	0	Balach.	44.5	0.0088	0.441	0
	Proposed	8.8	0.0028	0.424	0	Proposed	8.7	0.0028	0.426	0

Results report the efficiency (number of iterations and runtime (seconds)), registration error (RE) (mm), and instability (% of trials) of the GTLS registration method proposed in this paper compared to the closed-form isotropic solution[32] and the prior GTLS methods of Estépar et al.[55] and Balachandran and Fitzpatrick.[61] The tests are binned according to the magnitude of initial misalignment in translation (mm) and rotation (degrees); each bin represents average values measured over 1000 randomized trials.

dences following each registration. This value is reported as the *registration error* (RE).

4.5.1.1 Experiment 1A: Anisotropic Noise in Both Sets of Points

The results of Experiment 1A, which has anisotropic noise in both sets of points, are presented in Table 4.1. In this study, two intervals of translational misalignment were investigated, namely $[10, 20]$ mm and $[90, 100]$ mm, along with five cases of rotational misalignment, which as a group covered the entire $[0, 180]$ degrees interval.

As seen in the results, the proposed Gauss-Newton-based method achieves lower registration error than all compared methods across all test cases. The proposed method also maintains consistent registration error across all misalignments studied, achieving significant improvement with respect to the isotropic solution in every case. In contrast, the prior anisotropic methods of Estépar et al. and of Balachandran and Fitzpatrick worsen in accuracy as rotational misalignment increases and tend to provide larger registration errors than even the isotropic solution for rotational misalignments on the interval $[45, 90]$ degrees and beyond.

The proposed method's runtime is also several times more efficient than the other anisotropic solutions; computing a solution requires many fewer iterations (4-9) compared to the methods of Estépar et al. (15-28) and Balachandran and Fitzpatrick (28-45). Note that the iteration count for the method of Estépar et al. is reported as the total number of evaluations of its inner loop, which is where the vast majority of computation takes place for that method.

Another significant observation regarding the results of Experimant 1A is that the proposed method and that of Balachandran and Fitzpatrick are stable under all conditions tested, whereas the method of Estépar et al. encounters frequent instability for large rotational misalignment, with the portion of unstable trials reaching 60% at the largest rotation interval of $[150, 180]$ degrees.

For the method of Balachandrian and Fitzpatrick, the increase in registration error with respect to rotation is understood to result from the assumption of a constant

noise covariance as described earlier. However, it is not clear why the method of Estépar et al. exhibits a similar issue. To shed more light on this and on the issue of instability we include a third study (Experiment 1C in this section).

4.5.1.2 Experiment 1B: Anisotropic Noise in One Set of Points

Table 4.2 presents the results of Experiment 1B, which incorporates anisotropic noise in only one point set and isotropic noise in the other. In this study, translational misalignment is limited to a large interval of $[90, 100]$ mm, while the test cases for rotational misalignment remain unchanged. The trials for this experiment are conducted twice: once with and once without initializing the anisotropic methods to the isotropic noise solution; the other test conditions (exact point sets, noise, etc.) remain identical between the two types of trials.

As seen in Table 4.2, the outcome is similar to the earlier study, with the most notable difference being that the method of Balachandran and Fitzpatrick computes an equally accurate registration as the proposed method, which confirms that the high registration errors encountered for this method in the prior study resulted from its assumption of anisotropic noise in only one point set. Concerning the method of Estépar et al., the increase in registration error with respect to rotational misalignment remains, which indicates a different source of error for that method.

It is also interesting to note, concerning the method of Estépar et al., that although the occurrence of unstable trials is reduced by initialization to the isotropic solution,

Table 4.2: Experiment 1B: corresponding point set registration results with anisotropic noise present in one set of points.

	Without Isotropic Initialization					With Isotropic Initialization				
Rot	**Alg**	**Iter**	**Time**	**RE**	**Inst**	**Alg**	**Iter**	**Time**	**RE**	**Inst**
$[0, 15]$	Isotropic	1.0	0.0001	0.349	0	Isotropic	-	-	-	-
	Estépar	14.6	0.0100	0.332	0	Estépar	10.4	0.0080	0.332	0
	Balach.	32.9	0.0068	0.333	0	Balach.	14.8	0.0039	0.332	0
	Proposed	3.7	0.0011	0.332	0	Proposed	2.9	0.0012	0.332	0
$[15, 45]$	Isotropic	1.0	0.0001	0.347	0	Isotropic	-	-	-	-
	Estépar	17.1	0.0115	0.338	0	Estépar	10.8	0.0076	0.338	0
	Balach.	37.5	0.0076	0.330	0	Balach.	14.6	0.0036	0.329	0
	Proposed	4.2	0.0012	0.330	0	Proposed	2.9	0.0011	0.330	0
$[45, 90]$	Isotropic	1.0	0.0001	0.341	0	Isotropic	-	-	-	-
	Estépar	18.7	0.0125	0.352	0	Estépar	10.8	0.0077	0.352	0
	Balach.	40.2	0.0081	0.325	0	Balach.	14.4	0.0037	0.325	0
	Proposed	5.0	0.0013	0.325	0	Proposed	2.9	0.0011	0.325	0
$[90, 150]$	Isotropic	1.0	0.0001	0.345	0	Isotropic	-	-	-	-
	Estépar	22.6	0.0149	0.366	2	Estépar	11.8	0.0082	0.365	0
	Balach.	41.9	0.0084	0.330	0	Balach.	14.6	0.0037	0.330	0
	Proposed	6.1	0.0015	0.330	0	Proposed	2.9	0.0011	0.330	0
$[150, 180]$	Isotropic	1.0	0.0001	0.350	0	Isotropic	-	-	-	-
	Estépar	26.7	0.0173	0.373	60	Estépar	16.2	0.0109	0.385	10
	Balach.	44.5	0.0089	0.333	0	Balach.	14.5	0.0037	0.333	0
	Proposed	8.5	0.0018	0.333	0	Proposed	2.9	0.0011	0.333	0

Results report the efficiency (number of iterations and runtime (seconds)), registration error (RE) (mm), and instability (% of trials) of the GTLS method proposed in this paper compared to the closed-form isotropic solution[32] and the prior GTLS methods of Estépar et al.[55] and Balachandran and Fitzpatrick.[61] The tests are binned according to the magnitude of initial misalignment in rotation (degrees), with all bins having a translational misalignment in the range $[90, 100]$ mm; each bin represents average values measured over 1000 randomized trials.

the problem of instability yet remains.

The proposed method remains largely more efficient than the other anisotropic methods, both with and without initialization to the isotropic solution. It is interesting to note that initialization to the isotropic solution approximately halves the runtime of the method by Balachandran and Fitzpatrick whereas the runtime for the proposed method is reduced by much less, even though the relative decrease in the number of iterations for each method is similar. This observation indicates that the proposed method has low computational complexity beyond that of computing the initial covariance decompositions (recall for this study that the effective covariances are assumed to be constant), whereas the method of Balachandran and Fitzpatrick retains significantly more overhead per iteration following the initial covariance decompositions. Isotropic initialization also significantly reduces the runtime of the method by Estépar et al. to a little more than half its value otherwise.

4.5.1.3 Experiment 1C: Rotational Alignment with Anisotropic Noise

Table 4.3 presents the results of the final study in this series, Experiment 1C, which is intended to further investigate the registration error and instability issues that are encountered by the method of Estépar et al. as the magnitude of rotational misalignment increases. This study evaluates only the rotational component of their method. That is, comparison is made concerning the GTLS rotational estimation method of

Table 4.3: Experiment 1C: Rotation-only registration results for corresponding point sets with anisotropic noise present in both sets of points.

Rot	Alg	Iter	Time	RE	Inst
$[0, 15]$	Isotropic	1.0	0.0000	0.304	0
	Kanatani	4.0	0.0025	0.279	0
	Proposed	3.8	0.0013	0.278	0
$[15, 45]$	Isotropic	1.0	0.0000	0.292	0
	Kanatani	4.0	0.0025	0.283	0
	Proposed	4.4	0.0015	0.269	0
$[45, 90]$	Isotropic	1.0	0.0000	0.295	0
	Kanatani	4.0	0.0025	0.313	0
	Proposed	5.1	0.0017	0.271	0
$[90, 150]$	Isotropic	1.0	0.0000	0.292	0
	Kanatani	4.5	0.0028	0.323	0
	Proposed	6.3	0.0020	0.265	0
$[150, 180]$	Isotropic	1.0	0.0000	0.286	0
	Kanatani	6.1	0.0038	0.360	10
	Proposed	8.7	0.0028	0.263	0

Results report the efficiency (number of iterations and runtime (seconds)), registration error (RE) (mm), and instability (% of trials) of the GTLS method proposed in this paper (modified to compute only rotation) compared to the closed-form isotropic solution[32] and the prior GTLS rotation estimation method of Ohta and Kanatani.[62] The tests are binned according to the magnitude of initial misalignment in rotation (degrees) with translational misalignment being zero; each bin represents average values measured over 1000 randomized trials.

Ohta and Kanatani[62] relative to the proposed Gauss-Newton-based GTLS method and relative to the rotation computed under an isotropic noise assumption. For this comparison, the proposed method and the isotropic solution were modified to estimate only parameters of rotation, assuming translation to be zero. This study was conducted in similar fashion as Experiment 1A, except that only rotational misalignment was applied between the two point sets.

As seen in Table 4.3, the method of Ohta and Kanatani exhibits the same increase in registration error relative to rotational misalignment as encountered by the method of Estépar et al. Further, the rotation estimation exhibits similar instability under

large rotational misalignment. This indicates that a source of error and instability for the method of Estépar et al. lies in the rotation estimation component.

The rotation estimation method in this case uses a quaternion parameterization for rotation that is optimized by applying the renormalization method of Kanatani.[63] Matei and Meer[64] have presented a technique called heteroscedastic errors-in-variables (HEIV) estimator, which is closely related to the renormalization method of Kanatani, in which they present a similarly parameterized method for computing the full rigid-body alignment of two point sets under anisotropic noise. In their work, they point out a discontinuity in the quaternion representation for rotation that produces instability for rotational misalignments close to ± 180 degrees. It is possible that the rotation estimation method of Kanatani suffers from a similar issue, though we do not verify this further within this work.

4.5.2 Experiment 2: Registering a Mesh Model without Outliers

In this study, we evaluate the performance of the IMLP algorithm for registering a model shape represented by a triangular mesh. The experiment is divided into two sub-experiments (Experiments 2A and 2B) in order to evaluate the algorithm's performance under different magnitudes of shape misalignment. The shape being registered in both cases is a human pelvis model segmented from CT imaging to form

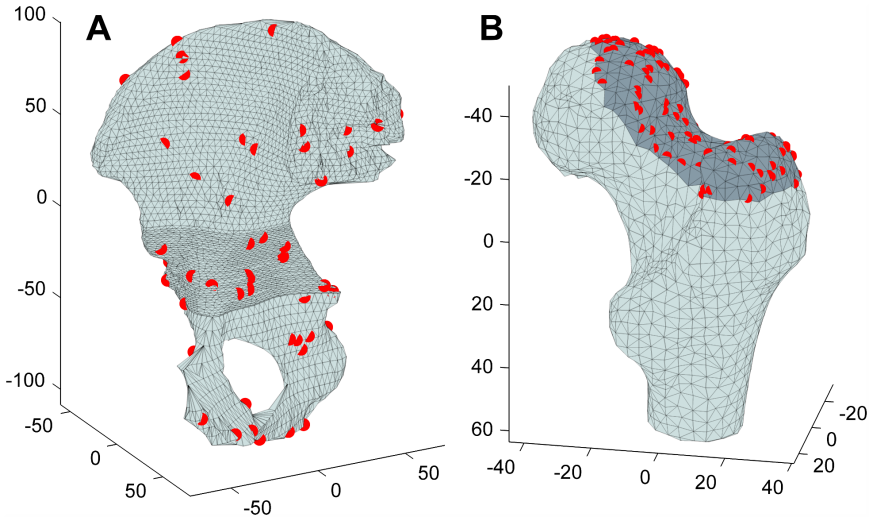

Figure 4.2: Human pelvis- and femur-bone meshes used in the registration studies. The red points represent a typical randomly generated data shape as sampled from the mesh surface. (A) pelvis mesh used in registration Experiments 2-5; (B) femur mesh used for the sub-shape registration study of Experiment 6, where points forming the data shape are generated from the darkly shaded region of the mesh.

a surface mesh (Figure 4.2A).

The experiments are conducted by randomly generating a set of 100 noisy points from the mesh surface to form a data shape and applying a random misalignment between the data shape and the mesh. The data points are then registered back to the mesh, which serves as the model shape. This approach enables accurate assessment of the registration error under varying noise conditions, since both the ground-truth alignment and the generative noise models are known.

Nine different test cases were studied to evaluate nine different noise models defined in Table 4.4, which specifies the standard deviation of multivariate Gaussian noise generated in the surface-normal vs. surface-parallel directions at each data point. Thus, the noise applied to each data point was conditioned relative to the

Table 4.4: Generative noise models (test cases) used in the randomized registration trials of Experiments 2-5.

Test Case	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Surface-Normal Std. Dev. (mm)	0.5	1.0	2.0	1.0	2.0	2.0	0.5	1.0	0.5
Surface-Parallel Std. Dev. (mm)	0.5	1.0	2.0	0.5	1.0	0.5	1.0	2.0	2.0

This table defines the standard deviation of noise generated in the surface-normal and surface-parallel directions for the data shapes used in Experiments 2-5.

orientation of the surface at that point. The first three test cases apply isotropic noise, the next three tests cases apply anisotropic noise of high surface-normal variance, and the final three test cases apply anisotropic noise of high surface-parallel variance, each in order of increasing magnitude of variance and increasing anisotropy. Within each of the nine test cases, 300 randomized registration trials were conducted, each involving different randomly generated data points, noise, and misalignment. All algorithms tested were executed once per trial under exactly the same test conditions (i.e., identical shape, noise, and misalignment) as generated for that trial.

Registration errors were measured by randomly sampling a set of 100 non-noisy points from the mesh surface to be used in validation. Following registration, the average distance between the registered and known ground-truth positions of the validation points was measured and recorded as the *target registration error (TRE)*.

The TREs of the registration trials are reported in several ways for each algorithm and test case. In the first method, an average TRE is computed from the set of registration trials that have a TRE below 10 mm. The registration trials that satisfy this condition are referred to as the "successful" registrations, and those that do not

satisfy this condition are referred to as the "failed" registrations. Failed registrations are excluded from the average TRE calculation in order to obtain an accurate assessment of registration accuracy for trials that converge near the correct solution. If the failed trials were not excluded, it would be possible for a few very poor outcomes to largely impact the total average, and thereby mask the true registration accuracy. For this method, the registration *failure rates* and the approximate standard deviations of the TRE averages are also reported in addition to the average TREs of the successful trials. The standard deviation of a TRE average is calculated by computing the standard deviation of the values used to compute the TRE average and then dividing by the square root of the number of values (i.e., dividing by the square root of the number of successful registration trials for that average). This is the procedure for calculating the standard deviation of a sample mean for independent identically distributed Gaussian data.[66] Since the histograms of TREs of the successful registration trials reasonably resemble those of a Gaussian distribution in most cases, these standard deviation values may be considered to be close approximations. As a second method, the median and 95th percentile order statistic of the TREs are reported, which are inherently robust to outlying TRE values. As a third and final method, TRE histograms provide more complete detail concerning the distribution of the registration outcomes. Since many graphs would be required to report all outcomes in histogram form, this method is used for a subset of the registration test cases. These three reporting methods are used in this and the following experiments

of this chapter.

This experiment compares the algorithms of ICP,[4] IMLP, and the two variants IMLP-CP and IMLP-MD, which, as described in the introduction to Section 4.5, provide near comparison to the GTLS-ICP[55] and A-ICP[57] algorithms, respectively. For the IMLP algorithm (and variants), the measurement-noise covariances of the data points were set to the generative noise models defined in Table 4.4, while the measurement covariances for the model shape were set to zero, since no noise was added to the mesh. Because the mesh fully represents the continuous surface of the model shape, the surface-model covariances of the IMLP algorithm (and variants) were also set to zero, as these are intended for registering non-continuous (i.e., point-cloud) surface representations. Outlier detection was disabled by setting IMLP's (and variants') chi-square inverse CDF threshold (χ^2_{thresh}) to a large value.

This experiment was conducted twice for two different intervals of random initial misalignment: $[15, 30]$ mm translation and $[15, 30]$ degrees rotation (Experiment 2A) and $[30, 60]$ mm translation and $[30, 60]$ degrees rotation (Experiment 2B). These misalignments were generated along random directions of translation and along random axes of rotation.

Results from Experiment 2 are now discussed. The average TREs of the successful trials (trials with TRE < 10 mm) and the registration failure rates for each algorithm and test case are reported in Figures 4.3A (for Experiment 2A) and 4.3B (for Experiment 2B) and in Table 4.5, respectively. As seen in the figures, similar registration

accuracy is obtained for both levels of misalignment. As expected, the average TREs for the first three test cases involving isotropic noise are identical among all algorithms because, in this case, the noise model of IMLP reduces to the Euclidean-distance computations of the standard ICP algorithm. For the anisotropic noise test cases, the IMLP algorithm consistently achieves slightly improved or approximately equivalent registration accuracy compared to ICP. For this study, the Mahalanobis-distance-matching variant (IMLP-MD) computes approximately the same registration errors as IMLP, whereas the closest-point-matching variant (IMLP-CP) generally performs worse, even performing worse than ICP for the test cases involving a relatively higher variance of noise in the surface-normal direction. The error bars shown in Figure 4.3 (and in other figures throughout this chapter) are approximate standard deviations of the TRE averages. As seen in Table 4.5, the algorithms performed near equally in terms of registration failure rate with a maximum failure rate of 2% overall for Experiment 2B. Failure rates for Experiment 2A were 0% for all cases and are not shown in the table. The standard deviations of the TRE averages and the registration failure rates are computed as described in an earlier paragraph of this section.

Order statistics for the TRE outcomes, including the median and 95th percentile values, are shown in Figures 4.4A (for Experiment 2A) and 4.4B (for Experiment 2B). The order statistic outcomes closely parallel those of the average TRE outcomes as described for Figure 4.3. Finally, histograms of the TRE outcomes from the registration trials for test cases 4–6 (Table 4.4) of Experiment 2A are shown in Figure 4.5,

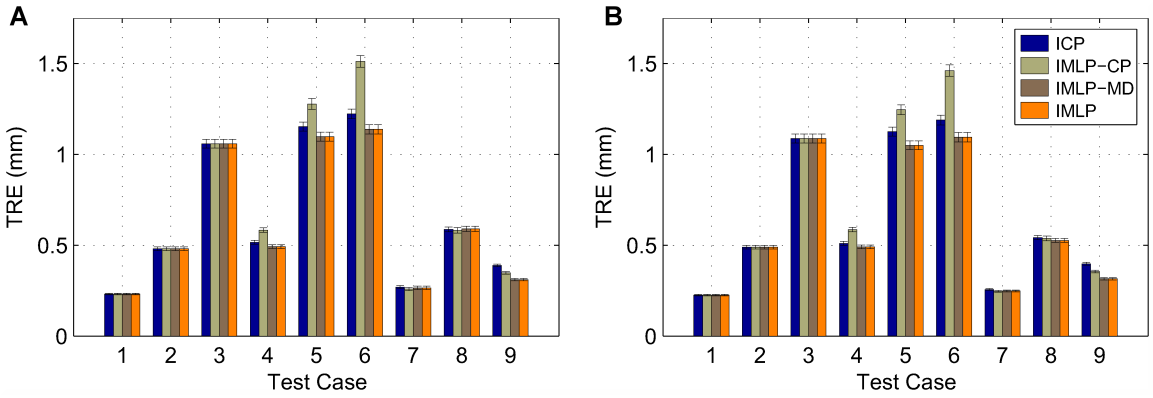

Figure 4.3: Experiment 2: average TREs of the successful registration trials (trials with TRE < 10 mm) for registering a data shape without outliers to a mesh representation of a model shape. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by (A) $[15, 30]$ mm (degrees) and (B) $[30, 60]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate noise in the data shape (Table 4.4). For each test case, 300 randomized trials were conducted, with successful registrations being used to compute an average TRE. The error bars show approximate standard deviations of the reported average TRE values. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to near comparisons of GTLS-ICP[55] and A-ICP[57] using IMLP-CP and IMLP-MD, respectively, which modify IMLP's most-likely-match criterion to that of closest-point and Mahalanobis-distance matching.

Table 4.5: Experiment 2: registration failure rates for registering a data shape without outliers to a mesh representation of a model shape.

	Failure Rate (%) by Test Case								
Alg	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
ICP	0.3	0.7	0.3	1.0	0.3	2.0	1.0	0.7	0.3
IMLP-CP	0.3	0.7	0.3	1.0	0.3	2.0	1.0	0.7	0.7
IMLP-MD	0.3	0.7	0.3	1.0	0.3	2.0	1.0	0.7	0.7
IMLP	0.3	0.7	0.3	1.0	0.3	2.0	1.0	0.7	0.7

Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees) in Experiment 2A and $[30, 60]$ mm (degrees) in Experiment 2B, and registered directly back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). For each test case, 300 randomized trials were conducted. This table reports the percent of unsuccessful registrations (TRE > 10 mm) for Experiment 2B. Registration failure rates for Experiment 2A were 0% across all algorithms and test cases and are not shown.

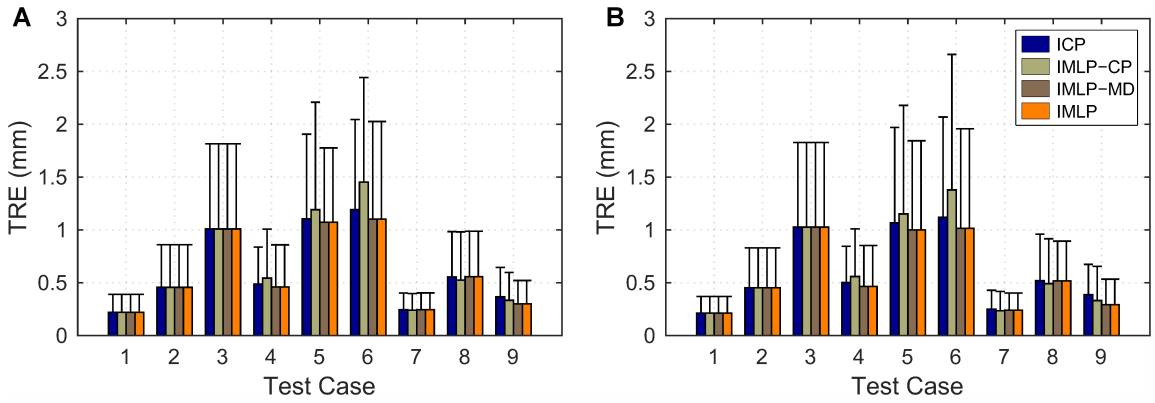

Figure 4.4: Experiment 2: the median and 95th percentile order statistics of the TRE outcomes for registering a data shape without outliers to a mesh representation of a model shape. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by (A) $[15, 30]$ mm (degrees) and (B) $[30, 60]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). For each test case, 300 randomized trials were conducted. The bar graphs and error bars show, respectively, the median and 95th percentile order statistics of the TRE outcomes for each test case. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to near comparisons of GTLS-ICP[55] and A-ICP[57] using IMLP-CP and IMLP-MD, respectively, which modify IMLP's most-likely-match criterion to that of closest-point and Mahalanobis-distance matching.

which show the distribution of TRE outcomes in greater detail.

Table 4.6 presents the runtime averages for the successful registrations within each experiment. Due to its higher complexity, IMLP has a runtime of approximately 3.5 times greater than ICP on average.

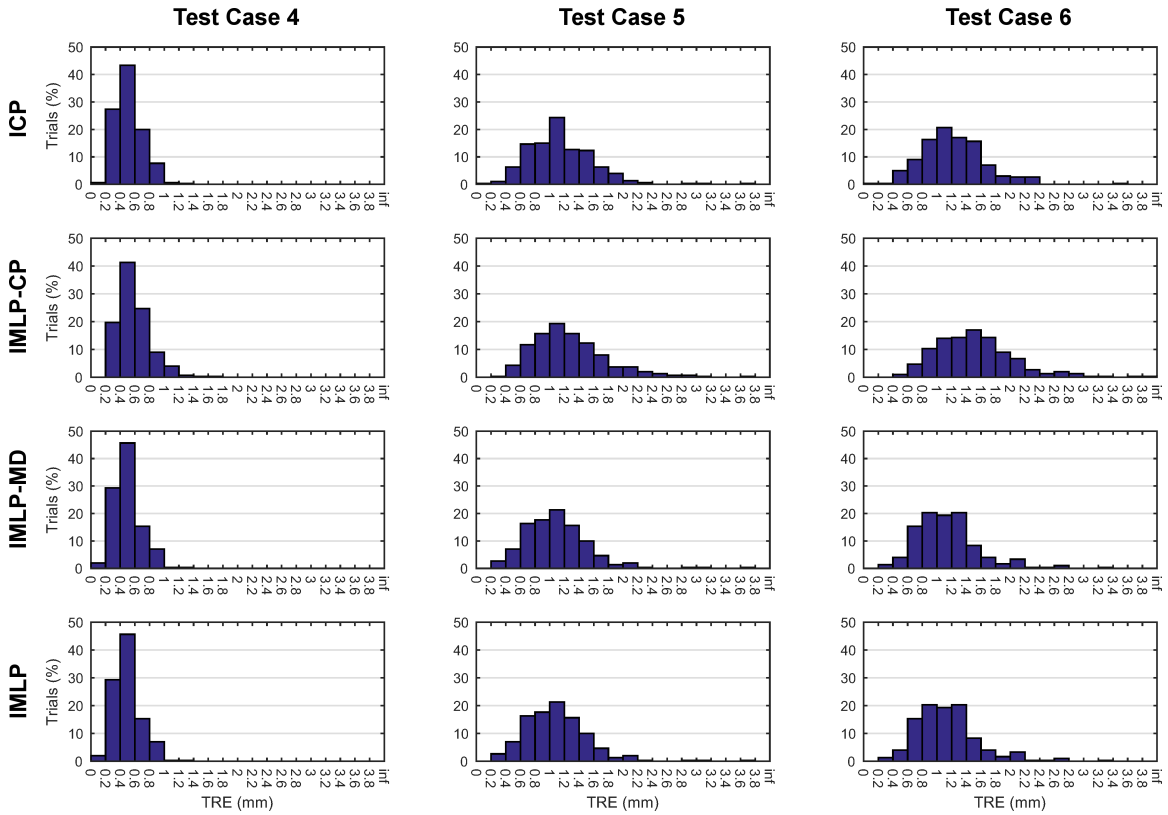

Figure 4.5: Experiment 2A: histograms of the TRE outcomes for registering a data shape without outliers to a mesh representation of a model shape under moderate misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to near comparisons of GTLS-ICP[55] and A-ICP[57] using IMLP-CP and IMLP-MD, respectively, which modify IMLP's most-likely-match criterion to that of closest-point and Mahalanobis-distance matching.

Table 4.6: Experiment 2: algorithm runtimes for registering a model shape represented by a mesh.

		Average Runtime (sec.) by Test Case								
Exp	**Alg**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
2A	ICP	0.086	0.094	0.109	0.107	0.119	0.115	0.087	0.097	0.095
	IMLP-CP	0.194	0.204	0.228	0.252	0.287	0.263	0.138	0.157	0.118
	IMLP-MD	0.270	0.299	0.323	0.389	0.437	0.401	0.226	0.249	0.219
	IMLP	0.352	0.377	0.407	0.499	0.543	0.569	0.258	0.239	0.202
2B	ICP	0.142	0.147	0.165	0.145	0.16	0.168	0.135	0.142	0.166
	IMLP-CP	0.285	0.286	0.322	0.346	0.351	0.343	0.221	0.222	0.196
	IMLP-MD	0.339	0.372	0.381	0.462	0.496	0.467	0.298	0.311	0.275
	IMLP	0.444	0.454	0.495	0.588	0.638	0.686	0.425	0.480	0.379

Average runtimes of successful registrations from Experiment 2 are reported, where 300 randomized trials were conducted for each test case. Each test case represents a different generative noise model (Table 4.4) for adding noise to the data shape. Results are reported for initial shape misalignments of $[15, 30]$ mm (degrees) (Experiment 2A) and $[30, 60]$ mm (degrees) (Experiment 2B).

4.5.3 Experiment 3: Registering a Mesh Model with Outliers

Experiment 3 follows the same procedure as Experiment 2, using the same model shape (the mesh of Figure 4.2A) and equivalent generative noise models (Table 4.4) and numbers of trials (300 per test case). However, in this study the data shape is corrupted with additional points added as outliers. This experiment is divided into two studies corresponding to initial shape misalignments within the ranges of $[15, 30]$ mm translation and $[15, 30]$ degrees rotation (Experiment 3A) and $[30, 60]$ mm translation and $[30, 60]$ degrees rotation (Experiment 3B). Each study is further divided into four sub-studies having varying percentages of outliers including 5%, 10%, 20%, and 30% outliers, which are referred to as sub-experiments i–iv, respectively. Since the purpose of this study is to evaluate the merit of IMLP's ability to handle outliers,

the two variant algorithms, IMLP-CP and IMLP-MD, (which use the same outlier mechanism) are not evaluated, while the Robust ICP[37] algorithm is added to the set of evaluated algorithms.

Outlier data was generated for each trial by randomly selecting points on the mesh surface and projecting each point outward from the mesh by a random distance generated uniformly from the interval $[10, 20]$ mm.

IMLP's outlier detection was enabled for these experiments by setting its chi-square inverse CDF threshold (χ^2_{thresh}) to the values of {7.81, 6.25, 4.64, and 3.66} for sub-experiments i–iv, respectively. These values correspond to chi-square inverse CDF probabilities of {0.95, 0.9, 0.8, 0.7}, which were chosen to directly correspond to the percentage of outliers in each test case.

For Robust ICP, the suggestion of its author was followed by relating the user parameter D, which is used to determine when a registration is good, to the resolution of the shape data. This was accomplished by computing the average distance between the triangle center points and that of their neighbors in the mesh (approximately 1.8 mm). The user parameter D^0_{max}, which controls the maximum tolerable match distance of the first iteration, was set to a large value in order to take all matches into consideration in the first iteration, which afforded the algorithm the best chance at computing a successful registration.

Results from Experiment 3 are now discussed. The average TREs of the successful trials (trials with TRE < 10 mm) and the registration failure rates for each algorithm

and test case are reported in Figures 4.6 (for Experiment 3A) and 4.7 (for Experiment 3B) and in Table 4.7, respectively. Figures 4.6 and 4.7 are each divided into four sub-figures (A–D), one for each level of outliers, corresponding to sub-experiments i–iv of Experiments 3A and 3B, respectively. The analysis to produce these results was conducted in the same manner as described for Experiment 2 (Section 4.5.2). As seen in the figures, IMLP widely outperforms ICP in terms of registration accuracy for all test cases and performs marginally better overall than Robust ICP for 5% and 10% outliers and much better than Robust ICP at higher percentages of outliers, where the TRE for Robust ICP approaches and even surpasses that of ICP. In contrast, IMLP's TRE remains fairly stable up to 20% outliers, beginning to increase at the 30% level. The registration failure rates shown in Table 4.7 for this study indicate that although ICP is the worst algorithm in terms of TRE, it also has the highest registration success rate. At the lesser misalignment range, the failure rates of all algorithms are below 1% for up to 10% outliers, with IMLP having marginally higher failure rates at the 20% outlier level. At 30% outliers, the failure rate of IMLP increases significantly, accompanied by a marginal increase in the failure rate of Robust ICP. For the large misalignment range, a different pattern emerges with Robust ICP exhibiting high failure rates across all outlier levels, whereas IMLP maintains low failure rates for up to 10% outliers. The failure rate of ICP increases marginally at the larger offset range yet still remains quite low.

Order statistics for the TRE outcomes, including the median and 95th percentile

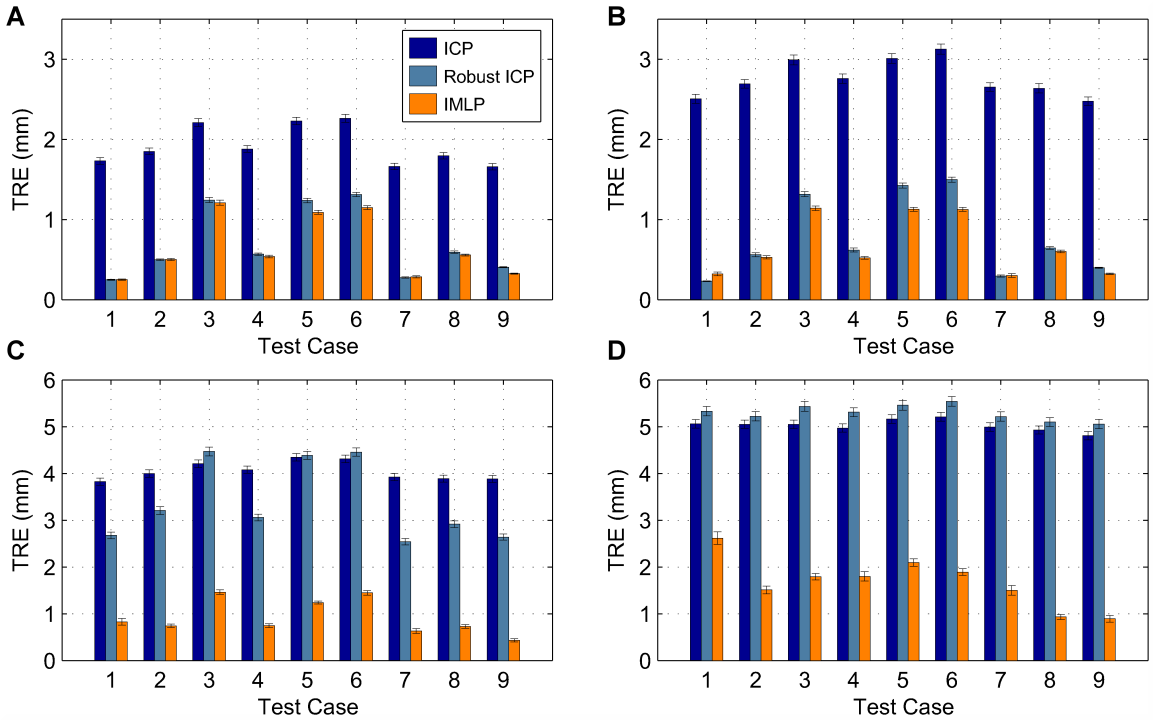

Figure 4.6: Experiment 3A: average TREs of the successful registration trials (trials with TRE $< 10\,\text{mm}$) for registering a data shape containing outliers to a mesh representation of a model shape under moderate misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]\,\text{mm}$ (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting (A) 5%, (B) 10%, (C) 20%, and (D) 30% of the data points. For each test case, 300 randomized trials were conducted, with successful registrations being used to compute an average TRE. The error bars show approximate standard deviations of the reported average TRE values. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

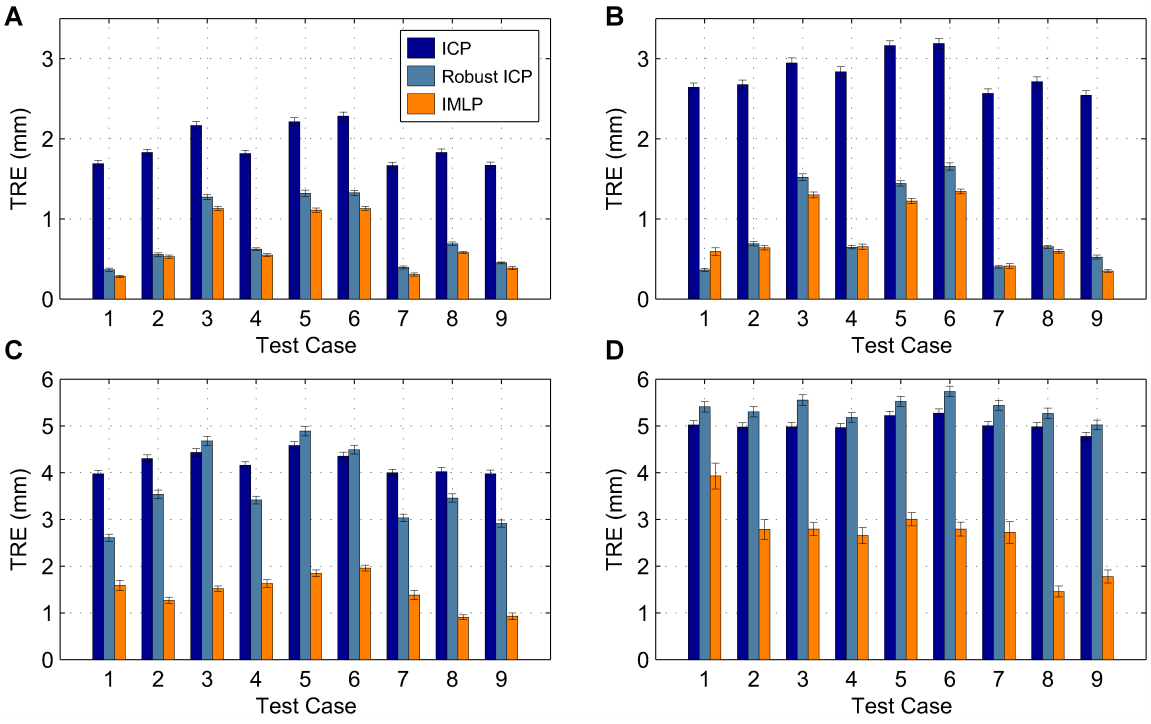

Figure 4.7: Experiment 3B: average TREs of the successful registration trials (trials with TRE $< 10\,\text{mm}$) for registering a data shape containing outliers to a mesh representation of a model shape under large misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[30, 60]\,\text{mm}$ (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting (A) 5%, (B) 10%, (C) 20%, and (D) 30% of the data points. For each test case, 300 randomized trials were conducted, with successful registrations being used to compute an average TRE. The error bars show approximate standard deviations of the reported average TRE values. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

Table 4.7: Experiment 3: registration failure rates for registering a data shape containing outliers to a mesh representation of a model shape.

			Failure Rate (%) by Test Case								
Exp	**Outliers**	**Alg**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
3A-i	5%	ICP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		Robust ICP	0.0	0.3	0.3	0.0	0.0	0.0	0.3	0.0	0.7
		IMLP	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.0	0.0
3A-ii	10%	ICP	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0
		Robust ICP	0.0	0.7	0.0	0.0	0.3	0.3	0.0	0.0	0.3
		IMLP	0.0	0.7	0.0	0.3	0.7	0.3	0.3	0.0	0.0
3A-iii	20%	ICP	0.3	0.3	0.7	0.0	0.0	0.3	0.3	0.3	1.0
		Robust ICP	1.0	0.7	1.0	0.7	0.0	1.3	1.3	0.7	1.0
		IMLP	4.0	2.7	4.0	4.3	2.0	1.7	3.7	2.0	3.7
3A-iv	30%	ICP	0.3	0.3	1.7	1.3	1.3	1.0	0.7	0.7	1.0
		Robust ICP	1.0	2.0	3.0	2.7	5.0	3.7	2.7	2.0	1.7
		IMLP	28.0	15.0	11.0	25.0	13.3	9.3	22.3	13.0	13.3
3B-i	5%	ICP	0.3	0.7	0.3	0.3	0.3	1.0	0.0	2.7	0.3
		Robust ICP	12.3	12.3	6.3	5.0	14.0	10.0	9.7	10.0	9.7
		IMLP	1.3	2.0	1.3	1.0	0.7	0.3	0.3	2.3	1.7
3B-ii	10%	ICP	0.7	1.3	0.3	0.3	0.7	1.0	1.0	1.3	1.0
		Robust ICP	11.7	7.7	9.3	11.7	10.0	12.7	8.7	7.0	8.3
		IMLP	3.0	2.0	3.0	5.7	1.3	3.3	2.3	2.3	2.3
3B-iii	20%	ICP	1.7	1.0	0.0	0.7	0.7	1.7	1.0	2.0	0.3
		Robust ICP	9.0	11.3	6.7	8.0	13.0	11.0	9.0	13.0	10.3
		IMLP	26.0	17.0	9.7	18.3	16.3	14.3	26.7	16.3	16.0
3B-iv	30%	ICP	2.7	2.0	2.3	2.0	2.7	1.7	1.3	1.7	2.3
		Robust ICP	12.7	17.3	12.7	12.3	13.7	14.3	10.3	11.3	12.7
		IMLP	75.0	65.3	47.0	62.7	52.3	53.0	75.3	55.3	57.7

Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees) in Experiment 3A and $[30, 60]$ mm (degrees) in Experiment 3B, and registered back to the mesh. The test cases represent the different noise models used to generate noise in the data shape (Table 4.4). Outliers were added to the data shape constituting 5% (-i), 10% (-ii), 20% (-iii), and 30% (-iv) of the data-shape points. For each test case, 300 randomized trials were conducted. This table reports the percent of unsuccessful registrations (TRE > 10 mm) for each test condition.

values, are shown in Figures 4.8 (for Experiment 3A) and 4.9 (for Experiment 3B), which follow a similar trend as described for the average TRE outcomes of Figures 4.6 and 4.7. That is, IMLP performs favorably compared to the ICP and Robust ICP algorithms, especially at outlier levels of 20% and above for the lesser misalignment range and at outlier levels of 20% and below for the greater misalignment range. For the outlier level of 30% in the greater misalignment range, IMLP has a worse median error than the other algorithms due to a large proportion of registration failures; this is an interesting test condition because although the median TRE of IMLP is larger than the other algorithms, the accuracy of IMLP remains superior for the trials that successfully registered. This observation is readily understood by referencing the histogram of outcomes for this experiment shown in Figure 4.17, where IMLP has a cluster of successful trials close to zero TRE and a large number of unsuccessful trials beyond 10 mm with few trials in between, whereas the other algorithms have TRE outcomes that are more evenly distributed between these two extremes. Concerning Figure 4.9, note that the large 95th percentile order statistic TRE for Experiment 3B-iii and the large median TRE for Experiment 3B-iv for the IMLP algorithm are consistent with the registration failure rates above 5% and 50%, respectively, as shown in Table 4.7. In other words, the TRE for the 95th percentile order statistic may be expected to be very large if the failure rate is above 5%; similarly, the median TRE may be expected to be very large if the failure rate is above 50%. This correlation is observed for the IMLP algorithm in Experiments

3B-iii and 3B-iv, respectively. The correlation between a failure rate above 5% and a high 95th percentile TRE is also observed for the Robust ICP algorithm for all four sub-studies in Experiment 3B. Finally, TRE histograms of the registration trials for Experiments 3A-i–iv and Experiments 3B-i–iv are shown in Figures 4.10–4.13 and Figures 4.14–4.17, respectively, for test cases 4–6 (Table 4.4), which show the distributions of the TRE outcomes in greater detail.

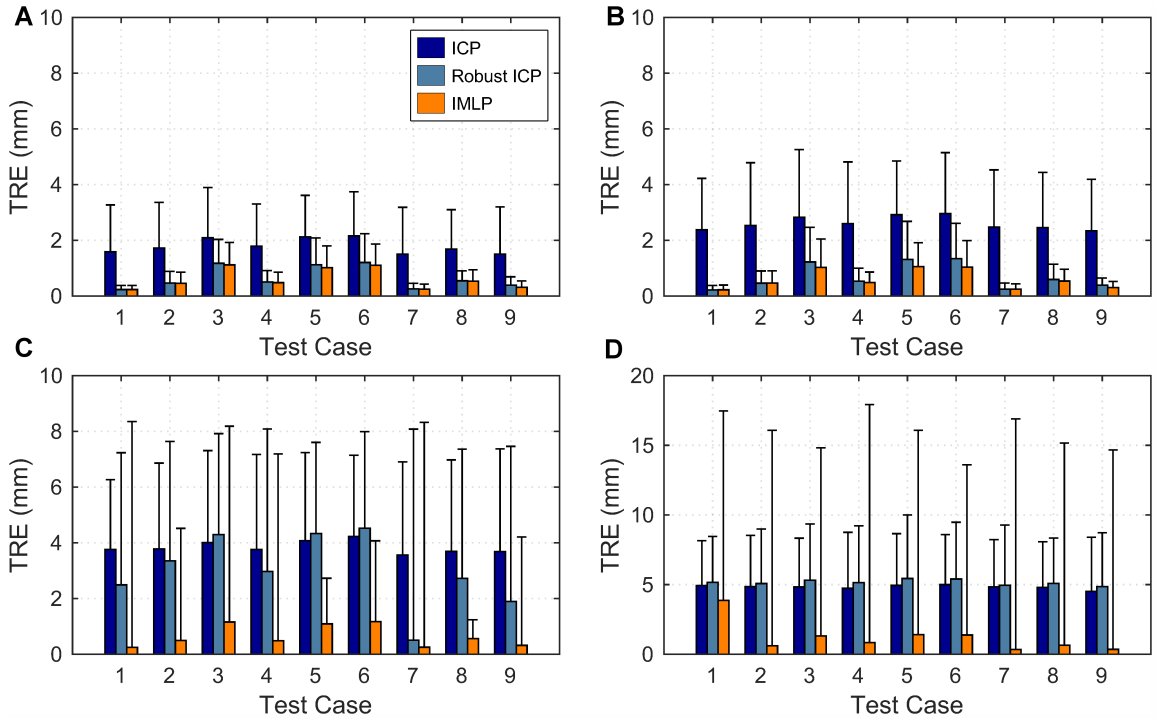

Figure 4.8: Experiment 3A: the median and 95th percentile order statistics of the TRE outcomes for registering a data shape containing outliers to a mesh representation of a model shape under moderate misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate noise in the data shape (Table 4.4). Outliers were added to the data shape constituting (A) 5%, (B) 10%, (C) 20%, and (D) 30% of the data points. For each test case, 300 randomized trials were conducted. The bar graphs and error bars show, respectively, the median and 95th percentile order statistics of the TRE outcomes for each test case. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

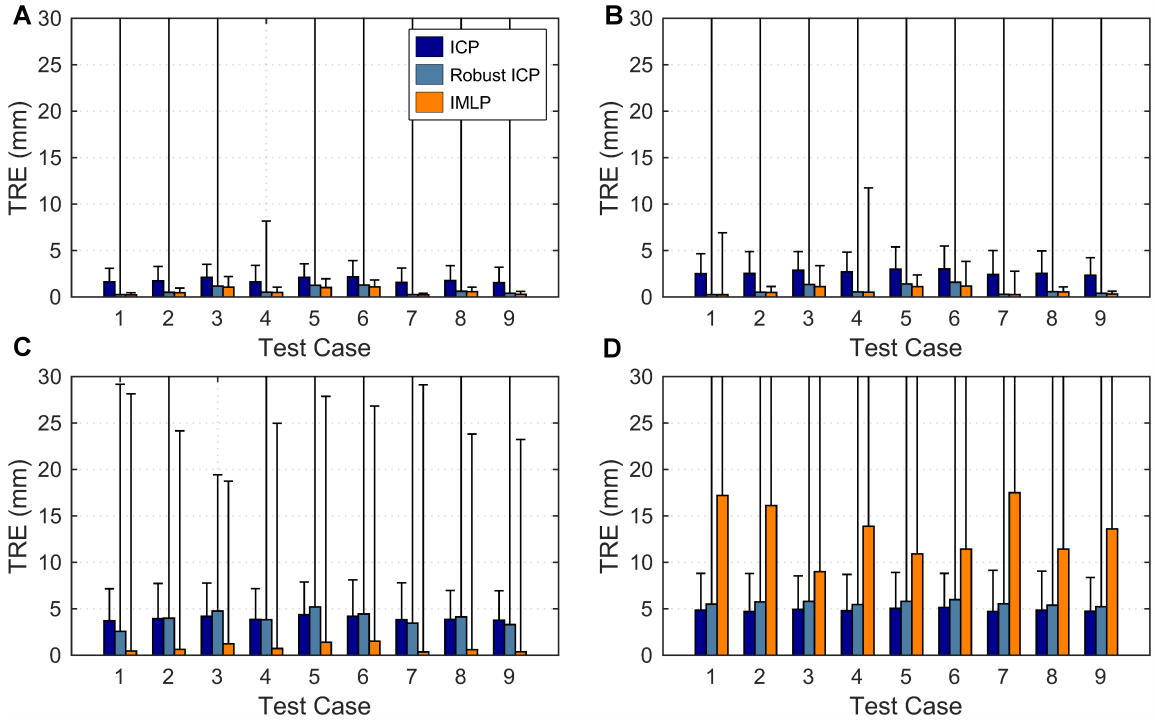

Figure 4.9: Experiment 3B: the median and 95th percentile order statistics of the TRE outcomes for registering a data shape containing outliers to a mesh representation of a model shape under large misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[30, 60]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate noise in the data shape (Table 4.4). Outliers were added to the data shape constituting (A) 5%, (B) 10%, (C) 20%, and (D) 30% of the data points. For each test case, 300 randomized trials were conducted. The bar graphs and error bars show, respectively, the median and 95th percentile order statistics of the TRE outcomes for each test case. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

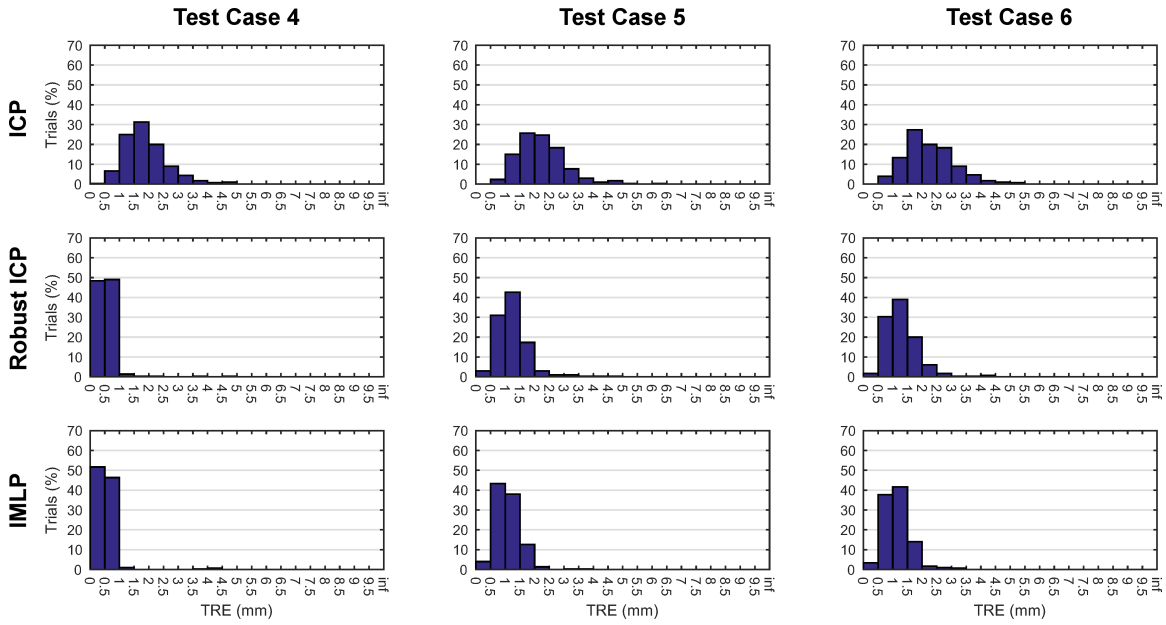

Figure 4.10: Experiment 3A-i: TRE histograms for the registration trials from test cases 4–6 for registering a data shape containing 5% outliers to a mesh model of a human pelvis (Figure 4.2A) under moderate misalignment. Data shapes were randomly generated from the mesh, misaligned by $[15, 30]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting 5% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

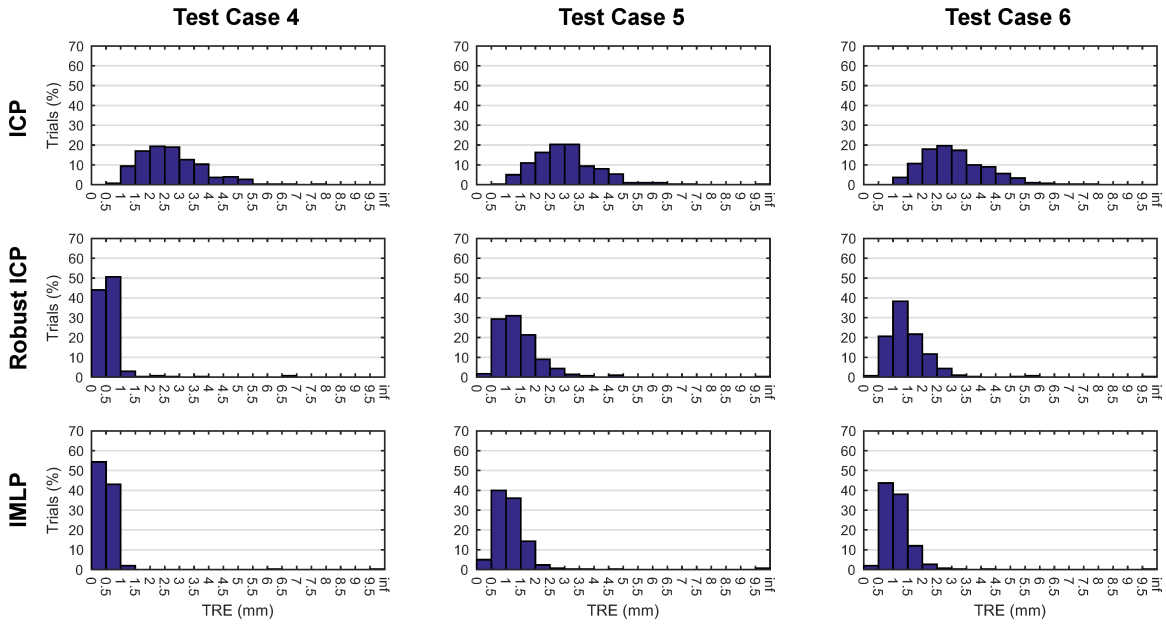

Figure 4.11: Experiment 3A-ii: TRE histograms for the registration trials from test cases 4–6 for registering a data shape containing 10% outliers to a mesh model of a human pelvis (Figure 4.2A) under moderate misalignment. Data shapes were randomly generated from the mesh, misaligned by $[15, 30]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting 10% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

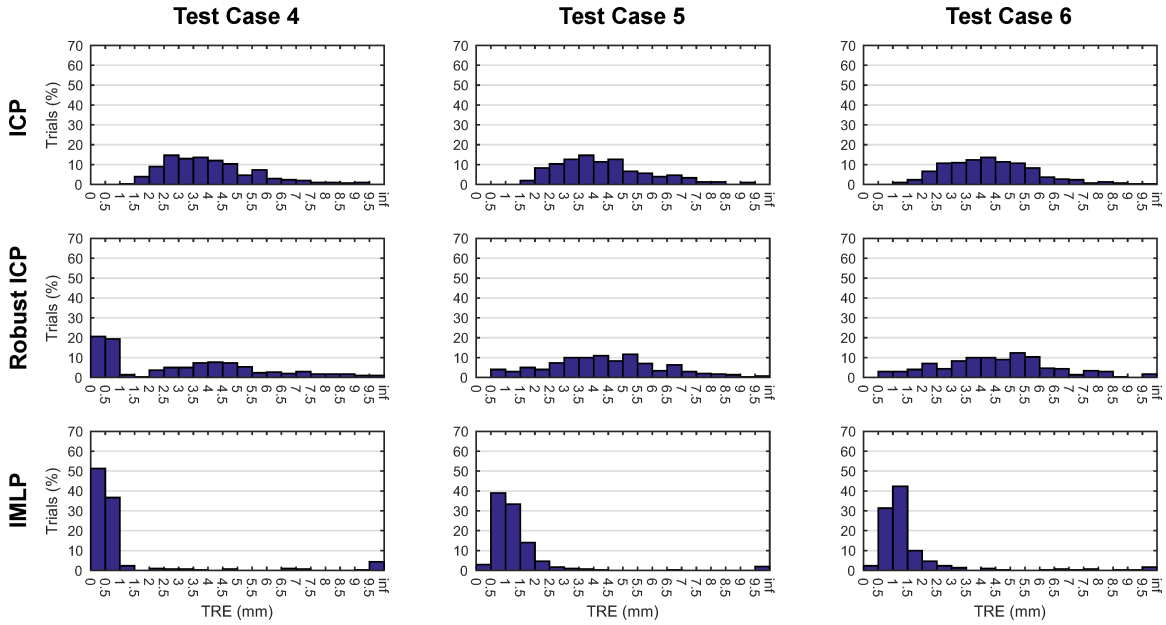

Figure 4.12: Experiment 3A-iii: TRE histograms for the registration trials from test cases 4–6 for registering a data shape containing 20% outliers to a mesh model of a human pelvis (Figure 4.2A) under moderate misalignment. Data shapes were randomly generated from the mesh, misaligned by $[15, 30]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting 20% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

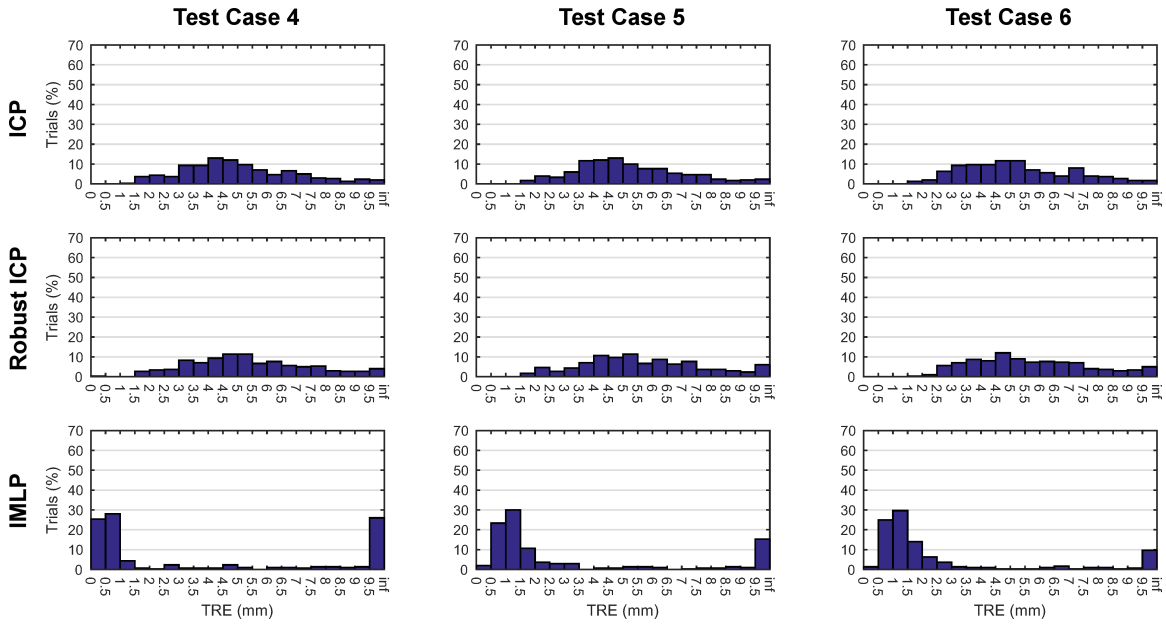

Figure 4.13: Experiment 3A-iv: TRE histograms for the registration trials from test cases 4–6 for registering a data shape containing 30% outliers to a mesh model of a human pelvis (Figure 4.2A) under moderate misalignment. Data shapes were randomly generated from the mesh, misaligned by $[15, 30]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting 30% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

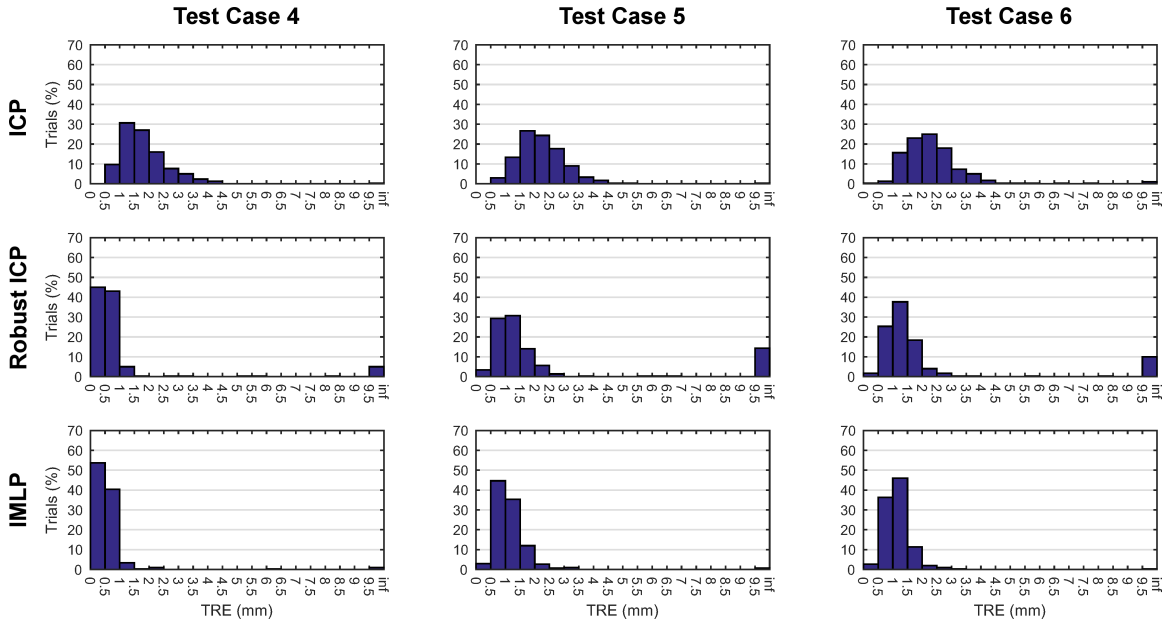

Figure 4.14: Experiment 3B-i: TRE histograms for the registration trials from test cases 4–6 for registering a data shape containing 5% outliers to a mesh model of a human pelvis (Figure 4.2A) under large misalignment. Data shapes were randomly generated from the mesh, misaligned by $[30, 60]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting 5% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

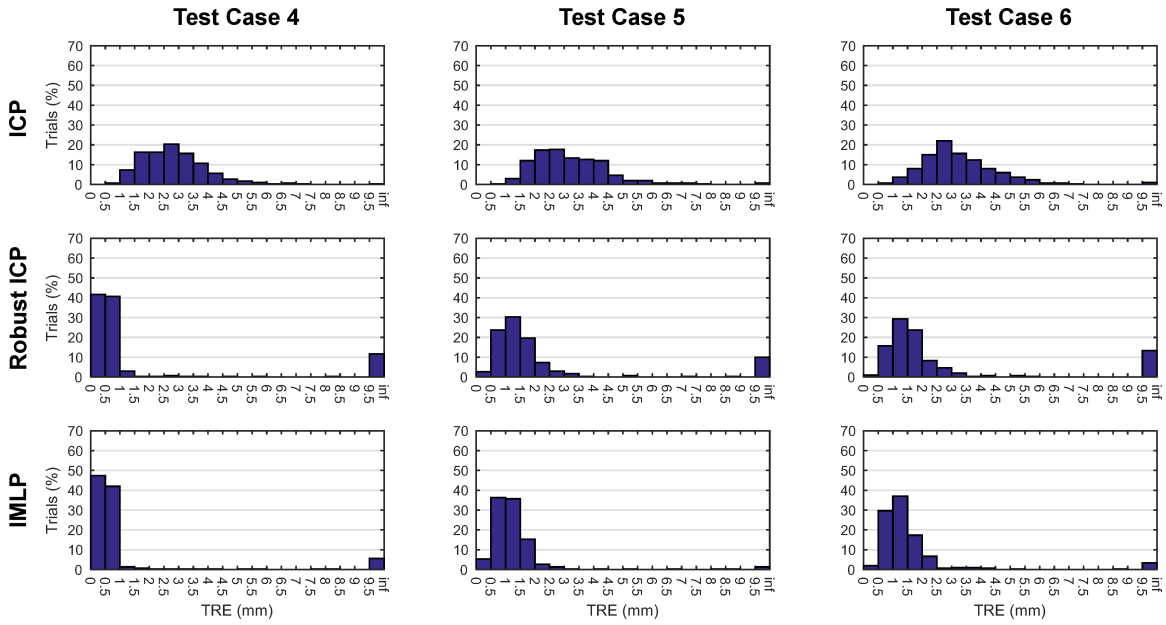

Figure 4.15: Experiment 3B-ii: TRE histograms for the registration trials from test cases 4–6 for registering a data shape containing 10% outliers to a mesh model of a human pelvis (Figure 4.2A) under large misalignment. Data shapes were randomly generated from the mesh, misaligned by $[30, 60]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting 10% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

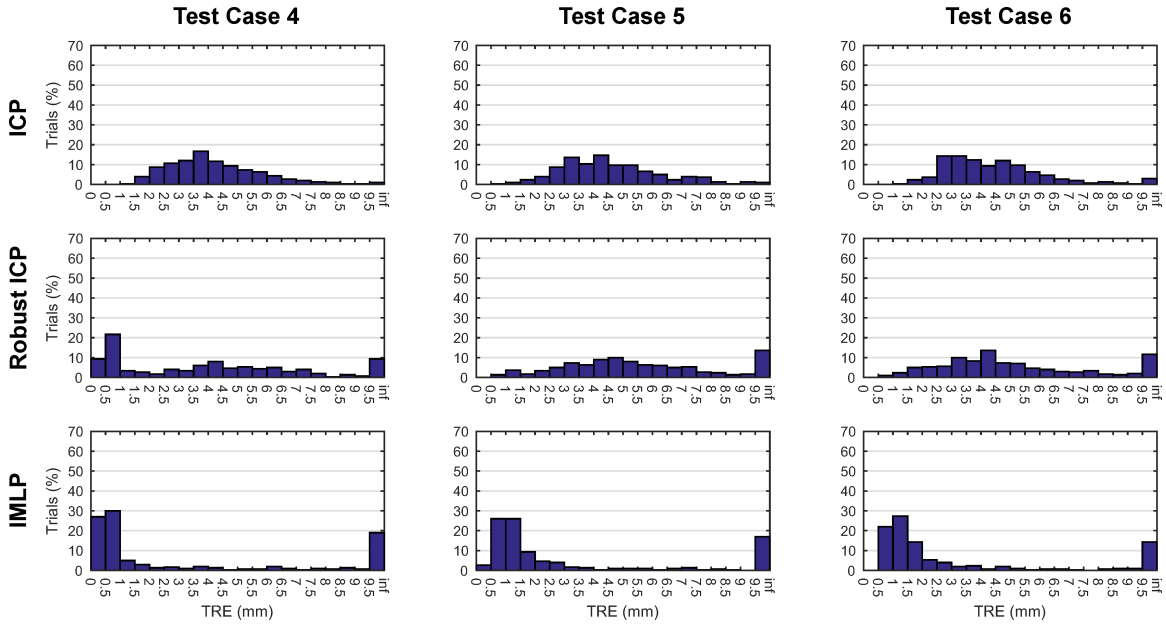

Figure 4.16: Experiment 3B-iii: TRE histograms for the registration trials from test cases 4–6 for registering a data shape containing 20% outliers to a mesh model of a human pelvis (Figure 4.2A) under large misalignment. Data shapes were randomly generated from the mesh, misaligned by $[30, 60]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting 20% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

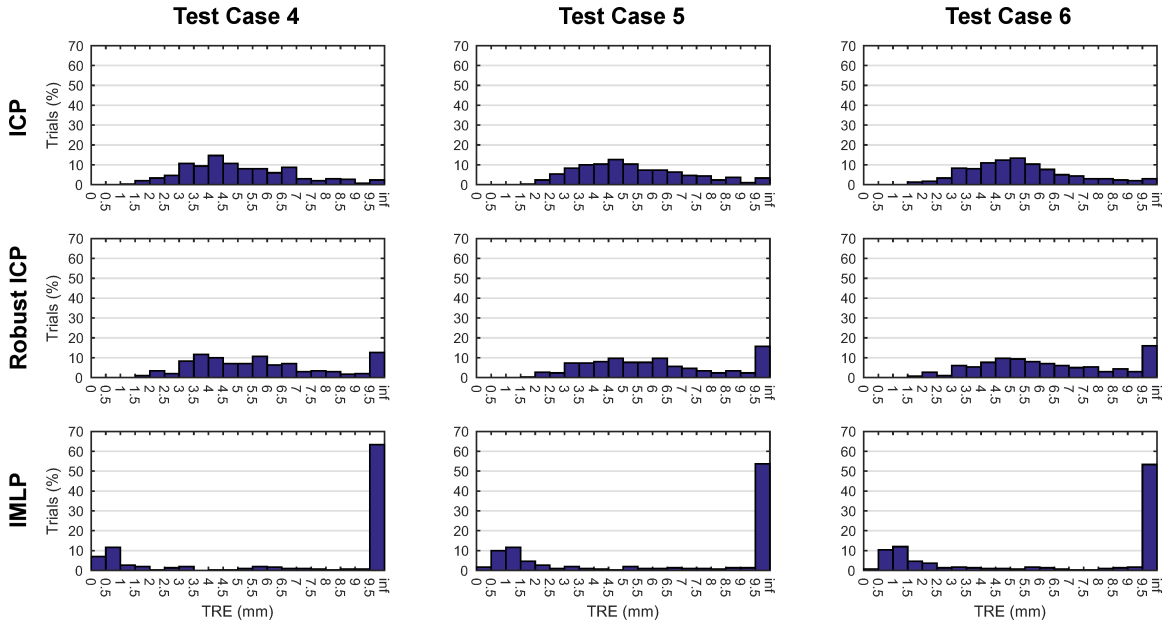

Figure 4.17: Experiment 3B-iv: TRE histograms for the registration trials from test cases 4–6 for registering a data shape containing 30% outliers to a mesh model of a human pelvis (Figure 4.2A) under large misalignment. Data shapes were randomly generated from the mesh, misaligned by $[30, 60]$ mm (degrees), and registered back to the mesh. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting 30% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP[4] and relative to a robust variant of ICP.[37]

4.5.4 Experiment 4: Registering a Point-Cloud Model without Outliers

In this study, the IMLP algorithm is evaluated for registering a model shape represented by a point cloud. Because the registration involves only point-cloud shapes, several additional algorithms can be compared. This experiment evaluates ICP,[4] GICP,[56] CPD,[46] IMLP, and the two variants IMLP-CP and IMLP-MD.

For this experiment, a dense point cloud composed of the center points of every triangle in the human pelvis mesh (Figure 4.2A) is used as the model shape. Besides this change, the test conditions for this experiment remain as described for Experiment 2 (Section 4.5.2) and are likewise divided into two parts representing different random misalignment intervals: $[15, 30]$ mm (degrees) (Experiment 4A) and $[30, 60]$ mm (degrees) (Experiment 4B).

In contrast to Experiment 2, the surface-model covariances of IMLP (and of its variants) are enabled in this study and defined to have standard deviations of 0.5 mm in the surface-normal direction and 5 mm in the surface-parallel directions. Recall that these covariances provide a local linear approximation of the unmeasured surface surrounding each sampled point in a point cloud in order to improve the accuracy of the registered shape alignment. The measurement-error covariances remain as defined in Experiment 2.

The GICP algorithm also employs a local surface model surrounding each sampled

point and uses its covariance matrices for this sole purpose. The covariance scaling parameter (ϵ) of GICP is set to 0.01, which is equal to the ratio of surface-normal vs. surface-parallel variances as used for IMLP. The surface-model covariances used for GICP are therefore equivalent to the surface-model covariances used for IMLP, because the optimizations performed by GICP do not change with respect to a global scaling of its covariances. We also tested GICP's default ϵ value of 0.001, but found 0.01 to provide higher accuracy in this study.

Outlier detection is disabled by setting the chi-square inverse CDF threshold of IMLP to a large value and setting the outlier weight of CPD to zero. The maximum match search distance of GICP is also set to a large value in order to not exclude any matches from consideration. Rigid-body transformation without scaling was selected as the CPD registration method, with the model-shape point cloud being used as the GMM centroids and the data-shape point cloud being used as the data points. This choice was found to be important, as reversing the roles of the data and model points for CPD produced substantially higher registration errors.

Results from Experiment 4 are now discussed. The average TREs of the successful trials (trials with TRE $< 10\,\text{mm}$) and the registration failure rates for each algorithm and test case are reported in Figures 4.18A (for Experiment 4A) and 4.18B (for Experiment 4B) and in Table 4.8, respectively. Similar results were obtained for both ranges of initial misalignment. As seen in the figures, IMLP achieves significantly better registration accuracy than any other algorithm across all test cases, with ex-

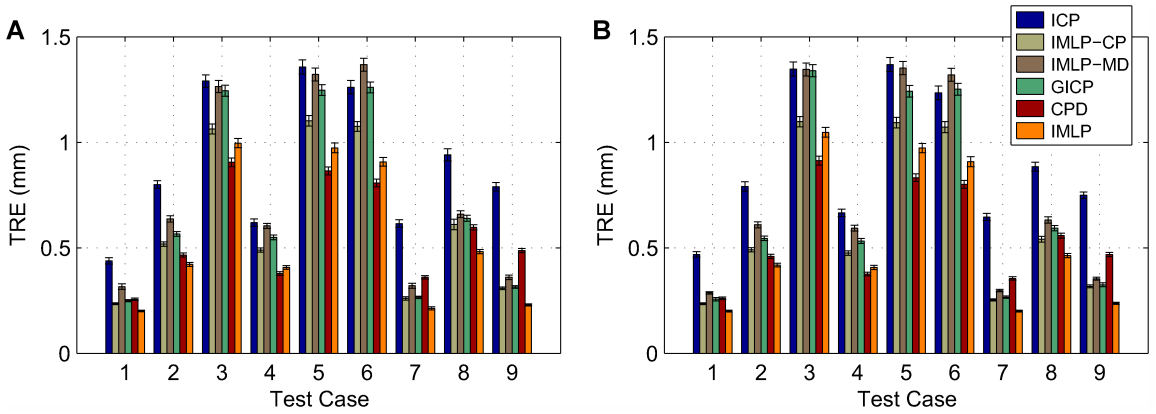

Figure 4.18: Experiment 4: average TREs of the successful registration trials (trials with TRE $< 10\,\text{mm}$) for registering a point-cloud representation of a model shape without data-shape outliers. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by (A) $[15, 30]\,\text{mm}$ (degrees) and (B) $[30, 60]\,\text{mm}$ (degrees), and registered back to a point-cloud representation of the model shape. The test cases represent different noise models used to generate data-shape noise (Table 4.4). For each test case, 300 randomized trials were conducted, with successful registrations being used to compute an average TRE. The error bars show approximate standard deviations of the reported average TRE values. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] and CPD,[46] as well as relative to near comparisons of GTLS-ICP[55] and A-ICP[57] using the two variants IMLP-CP and IMLP-MD, which modify IMLP's most-likely-match criterion to that of closest-point and Mahalanobis-distance matching, respectively.

ception of CPD compared to which IMLP achieves comparatively better accuracy in more than half of the test cases considered. Note that unlike Experiment 2A, in this experiment IMLP strongly outperforms ICP even for the initial test cases involving isotropic measurement noise. The reason for this stems from the surface-model covariances used to model non-represented surface regions surrounding each point-cloud sample point.

The advantage of IMLP's most-likely-point-matching criteria is particularly highlighted in comparison to the two variants of IMLP that evaluate modifications of this match criteria, i.e., closest-point matching (IMLP-CP) and Mahalanobis-distance

matching (IMLP-MD). IMLP achieves significantly, and in some cases substantially, higher accuracy than either of these variants for all test cases considered. Compared to GICP, IMLP also attains a notable accuracy improvement in all test cases, which further underscores the advantage of IMLP's most-likely-point-matching criteria and of its modeling of measurement error.

It is remarkable that the Mahalanobis-distance-matching criteria (IMLP-MD) has worse accuracy in this experiment than closest-point matching (IMLP-CP) and, for some test cases, shows no improvement over ICP. This is a surprising result, especially given that the reverse was true in Experiment 2 (Section 4.5.2), which involved registering to a mesh rather than to a point-cloud representation of the model shape.

Table 4.8 presents the registration failure rates of each algorithm for the large misalignment range of Experiment 4B. For the moderate misalignment range of Experiment 4A, no registration failures were indicated except for ICP, which had one failure in the second test case. As shown in the table, all algorithms achieve very low failure rates, with GICP being marginally higher than the others and CPD having the best performance with no registration failures.

Order statistics for the TRE outcomes, including the median and 95th percentile values, are shown in Figures 4.19A (for Experiment 4A) and 4.19B (for Experiment 4B). The order statistic outcomes closely parallel those of the average TRE outcomes as described for Figure 4.18. Finally, TRE histograms of the registration trials for test cases 7–9 (Table 4.4) of Experiment 4A are shown in Figure 4.20, which show

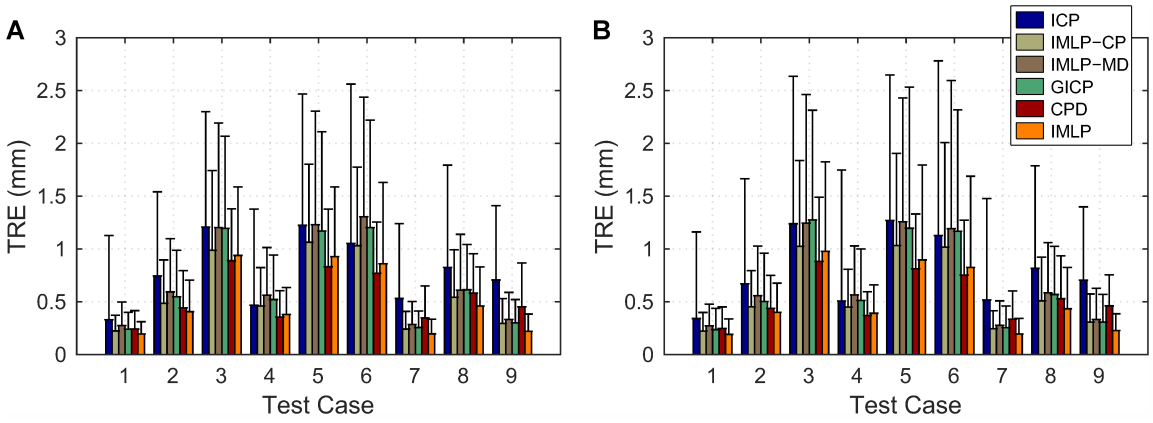

Figure 4.19: Experiment 4: the median and 95th percentile order statistics of the TRE outcomes for registering a point-cloud representation of a model shape without data-shape outliers. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by (A) $[15, 30]$ mm (degrees) and (B) $[30, 60]$ mm (degrees), and registered back to a point-cloud representation of the model shape. The test cases represent different noise models used to generate data-shape noise (Table 4.4). For each test case, 300 randomized trials were conducted. The bar graphs and error bars show, respectively, the median and 95th percentile order statistics of the TRE outcomes for each test case. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] and CPD,[46] as well as relative to near comparisons of GTLS-ICP[55] and A-ICP[57] using the two variants IMLP-CP and IMLP-MD, respectively, which modify IMLP's most-likely-match criterion to that of closest-point and Mahalanobis-distance matching.

the distribution of the TRE outcomes in greater detail.

Table 4.9 presents an average runtime comparison of each algorithm. ICP is the most efficient algorithm, with IMLP-CP coming second at approximately twice the runtime. While not shown in the table, the runtime of GICP is also on-par with that of IMLP-CP. The runtime of GICP is excluded from the table because it was executed within a Live Linux distribution running on a USB flash drive with persistent storage, which occasionally stuttered during execution causing inflated runtime averages. Although IMLP's runtime is approximately 9 times that of ICP in this study, IMLP is up to 60 times more efficient than CPD and 45 times more efficient

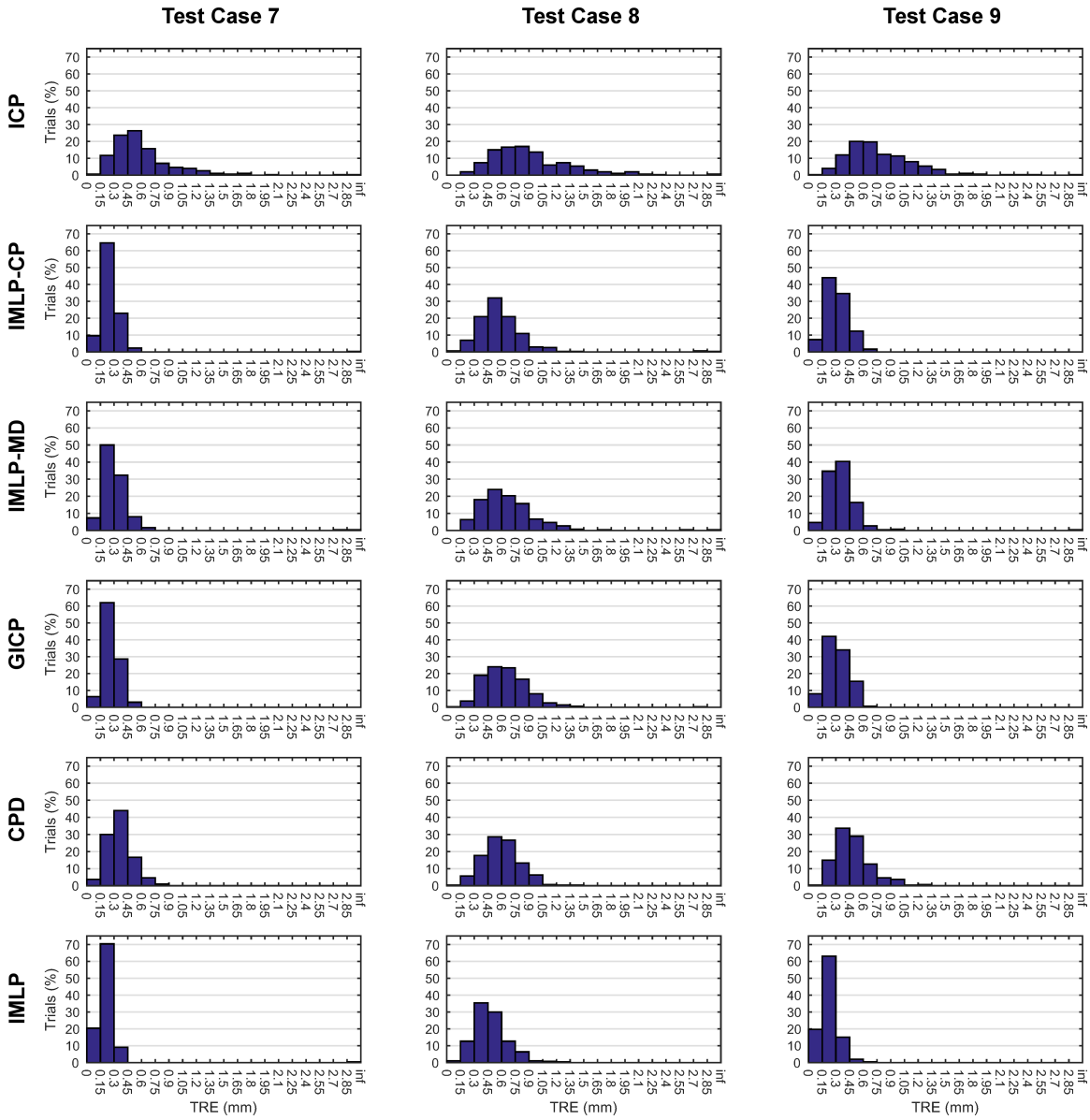

Figure 4.20: Experiment 4A: histograms of the TRE outcomes of test cases 7–9 for registering a point-cloud representation of a model shape under moderate misalignment without data-shape outliers. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees), and registered back to the mesh. The test cases represent a subset of the different noise models used to generate data-shape noise (Table 4.4). For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] and CPD,[46] as well as relative to near comparisons of GTLS-ICP[55] and A-ICP[57] using the two variants IMLP-CP and IMLP-MD, respectively, which modify IMLP's most-likely-match criterion to that of closest-point and Mahalanobis-distance matching.

Table 4.8: Experiment 4B: registration failure rates for registering a point-cloud model shape without outliers.

	Failure Rate (%) by Test Case								
Alg	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
ICP	0.3	0.7	0.3	0.7	0.3	2.0	0.7	0.3	0.3
IMLP-CP	0.3	0.7	0.3	0.7	0.3	2.0	0.7	0.3	0.3
IMLP-MD	0.3	0.7	0.3	0.7	0.3	2.0	1.0	0.3	0.3
GICP	1.3	1.3	0.7	2.7	3.7	2.0	2.0	1.3	1.7
CPD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
IMLP	0.3	0.7	0.3	0.7	0.3	2.0	1.0	0.3	0.3

Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees) in Experiment 4A and $[30, 60]$ mm (degrees) in Experiment 4B, and registered back to a point-cloud representation of the mesh. The test cases represent different noise models used to generate data-shape noise (Table 4.4). For each test case, 300 randomized trials were conducted. This table reports the percent of unsuccessful registrations (TRE > 10 mm) for each algorithm and test case of Experiment 4B. Failure rates for Experiment 4A (which are not shown in the table) were 0% for all algorithms and test cases, except for test case 2, where ICP incurred one registration failure.

on average. Using the input settings applied to CPD in this study, it was observed that CPD utilized 100% of both cores on the dual-core test platform, unlike the other algorithms which ran single-threaded. Thus, after normalizing for multithreading, IMLP is approximately two orders of magnitude more efficient than CPD.

In this study, the runtime difference between ICP and IMLP is greater than observed in Experiment 2A regarding a mesh-based model shape. This happens because the node search of the correspondence phase is simplified in this study by not having to compute the closest point on a triangle when computing the distance to a single datum in the PD tree. Although this provides a performance boost to both algorithms, the effect on ICP is much more pronounced since this computation occupies a greater percentage of ICP's overall runtime when it is required.

Table 4.9: Experiment 4: runtimes for registering a model shape represented as a point cloud.

		Average Runtime (sec.) by Test Case								
Exp	**Alg**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
4A	ICP	0.009	0.009	0.010	0.009	0.010	0.009	0.009	0.009	0.009
	IMLP-CP	0.015	0.016	0.019	0.016	0.020	0.019	0.015	0.017	0.015
	IMLP-MD	0.068	0.078	0.093	0.079	0.097	0.093	0.067	0.079	0.069
	GICP	-	-	-	-	-	-	-	-	-
	CPD (2 cores)	3.465	4.346	4.336	3.864	4.340	4.374	4.238	4.650	4.484
	IMLP	0.068	0.082	0.102	0.078	0.103	0.099	0.067	0.084	0.073
4B	ICP	0.013	0.013	0.013	0.013	0.013	0.013	0.012	0.012	0.013
	IMLP-CP	0.023	0.025	0.028	0.025	0.028	0.028	0.024	0.025	0.024
	IMLP-MD	0.100	0.109	0.126	0.112	0.127	0.129	0.100	0.109	0.099
	GICP	-	-	-	-	-	-	-	-	-
	CPD (2 cores)	3.584	4.408	4.490	4.279	4.327	4.545	4.378	4.731	4.874
	IMLP	0.101	0.111	0.134	0.115	0.136	0.133	0.103	0.118	0.106

Runtimes are reported as averages of the successful registrations from 300 randomized trials for each test case. The test cases represent different generative noise models (Table 4.4) applied to the data shape. Results are reported for initial shape misalignments of $[15, 30]$ mm (degrees) Experiment 4A and $[30, 60]$ mm (degrees) (Experiment 4B). Single-threaded implementations were used for ICP,[4] GICP,[56] IMLP, and the two IMLP variants IMLP-CP and IMLP-MD, whereas the implementation of CPD[46] was multi-threaded, making full utilization of the test platform's dual cores.

4.5.5 Experiment 5: Registering a Point-Cloud Model with Outliers

Experiment 5 follows a similar test scenario as Experiment 4 (Section 4.5.4), except that the data-shape point cloud is corrupted with additional points added as outliers. These outliers are generated in the same manner as described for Experiment 3 (Section 4.5.3). Like Experiment 3, this experiment is divided into two studies corresponding to initial shape misalignments within the ranges of $[15, 30]$ mm translation and $[15, 30]$ degrees rotation (Experiment 5A) and $[30, 60]$ mm translation and $[30, 60]$ degrees rotation (Experiment 5B). Each study is further divided into four

sub-studies for different percentages of outliers including 5%, 10%, 20%, and 30%, which are referred to as sub-experiments i–iv, respectively. As in Experiment 3, the two variants on IMLP are not included in this outlier study, whereas Robust ICP is added.

For the IMLP algorithm, the chi-square inverse CDF threshold (χ^2_{thresh}) is set according to the percentage of outliers as previously described in Experiment 3. Both the surface-model and measurement-error covariances are used in this study in the same manner as was described in Experiment 4. The user-defined parameters for Robust ICP are also configured as in Experiment 3. Following the lead of CPD's authors, the outlier weight is set to 0.5. In this case, the model-shape point cloud is assigned as the data points and the data-shape point cloud as the GMM centroids, which is the reverse of Experiment 3, because it was observed that this setting produced substantially lower registration error for the case of non-zero outlier weighting. Concerning the GICP algorithm, although a user-defined parameter is provided for limiting the match search distance, this mechanism is intended for partial shape registration rather than outlier handling. Although limiting the match search distance to 10 mm (in order to eliminate the outliers positioned 10–20 mm from the surface) improves the registration accuracy for some trials, this also causes registration failure in most cases. Thus, GICP is compared in this study by setting its maximum search distance to a large value and considering it to be a non-robust algorithm.

Results from Experiment 5 are now discussed. The average TREs of the successful

trials (trials with TRE < 10 mm) and the registration failure rates for each algorithm and test case are reported in Figures 4.21 (for Experiment 5A) and 4.22 (for Experiment 5B) and in Table 4.10, respectively. Figures 4.21 and 4.22 are each divided into four sub-figures (A–D) for each level of outliers, corresponding to sub-experiments (i–iv) of Experiments 5A and 5B, respectively. As in the prior studies, the TRE outcomes for both misalignment ranges are very similar. As seen in the figures, IMLP achieves large improvement in registration accuracy relative to the other algorithms for up to 20% outliers, even in comparison to CPD, which has a very effective outlier rejection capability. For the 30% outlier case, IMLP continues to provide accurate results and compares approximately equal to CPD. Compared to Robust ICP, IMLP is substantially more accurate in all test cases and frequently achieves less than half the registration error. As expected, ICP and GICP perform poorly, since they are non-robust techniques and do not include mechanisms to account for outliers. Robust ICP fairs much better than the non-robust methods for outlier compositions of 10% and below, but produces higher registration error than ICP for outlier levels above 10%. Table 4.10 shows the rate of registration failure for both ranges of misalignment. For moderate misalignment (Experiment 5A) all algorithms achieve very low failure rates for outlier compositions up to 20%, with exception of GICP which has high failure rate at 20% outliers and beyond. At 30% outliers, the failure rates of Robust ICP and IMLP moderately increase whereas the failure rates of ICP and CPD remain low with CPD achieving no registration failure. For large misalignment, the

failure rates of all algorithms are increased, with CPD again providing the best performance. IMLP is approximately on-par with CPD for outliers up to 10%. At 20% outliers, the failure rate of IMLP increases to a moderate 2 - 6.7%; at 30% outliers, the failure rate increases significantly to 12% and beyond. In contrast, the Robust ICP algorithm performs poorly across the board with an average failure rate above 10% for all percentages of outliers.

Order statistics for the TRE outcomes, including the median and 95th percentile values, are shown in Figures 4.23 (for Experiment 5A) and 4.24 (for Experiment 5B). The order statistic outcomes roughly parallel those of the average TRE outcomes as described for Figures 4.21 and 4.22. Note that unlike the average TRE value, the median TRE value for IMLP remains less than the median TRE value for CPD, even for the maximal case of 30% outliers. Finally, TRE histograms of the registration trials for test cases 7–9 (Table 4.4) are shown in Figures 4.25–4.28 (for Experiment 5A) and 4.29–4.32 (for Experiment 5B), which show the distributions of the TRE outcomes in greater detail.

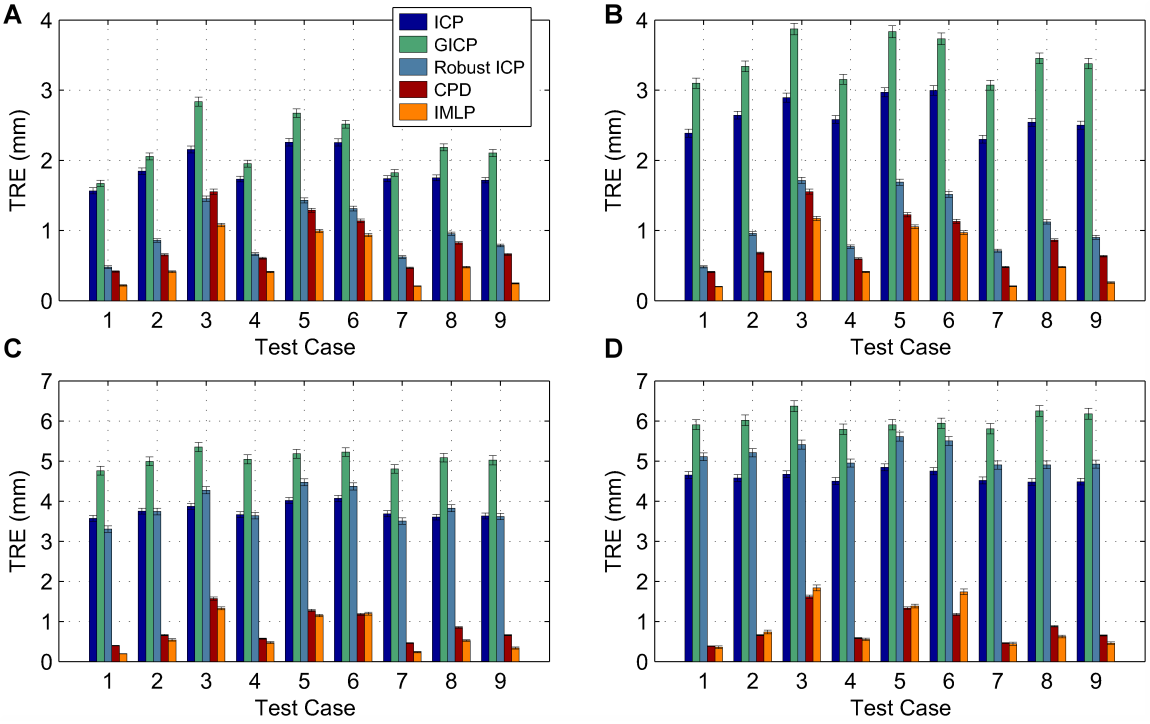

Figure 4.21: Experiment 5A: average TREs of the successful registration trials (trials with TRE $< 10\,\text{mm}$) for registering a data shape containing outliers to a point-cloud representation of a model shape under moderate misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]\,\text{mm}$ (degrees), and registered back to a point-cloud representation of the mesh. The test cases represent different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting (A) 5%, (B) 10%, (C) 20%, and (D) 30% of the data points. For each test case, 300 randomized trials were conducted, with successful registrations being used to compute an average TRE. The error bars show approximate standard deviations of the reported average TRE values. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

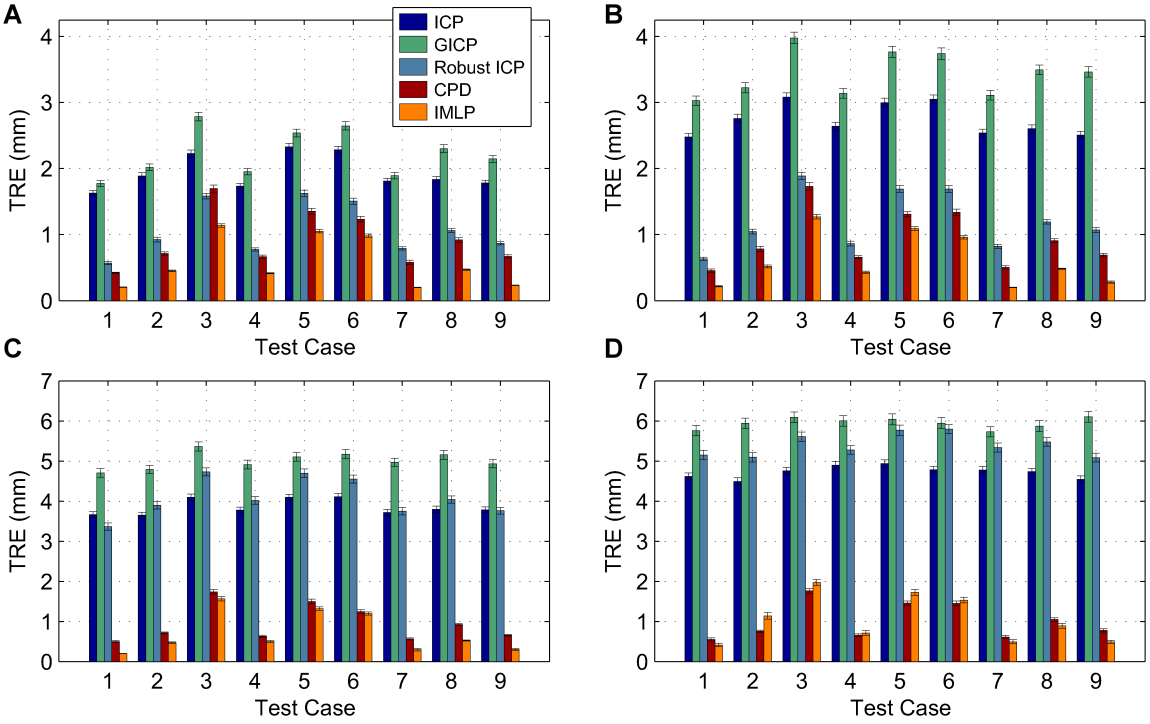

Figure 4.22: Experiment 5B: average TREs of the successful registration trials (trials with TRE $< 10\,\text{mm}$) for registering a data shape containing outliers to a point-cloud representation of a model shape under large misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[30, 60]\,\text{mm}$ (degrees), and registered back to a point-cloud representation of the mesh. The test cases represent different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting (A) 5%, (B) 10%, (C) 20%, and (D) 30% of the data points. For each test case, 300 randomized trials were conducted, with successful registrations being used to compute an average TRE. The error bars show approximate standard deviations of the reported average TRE values. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

Table 4.10: Experiment 5: registration failure rates for a point-cloud model shape with outliers.

			Failure Rate (%) by Test Case								
Exp	**Outliers**	**Alg**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
5A-i	5%	ICP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		GICP	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.3
		Robust ICP	0.0	0.3	0.0	0.0	0.0	0.0	0.3	0.3	0.3
		CPD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		IMLP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5A-ii	10%	ICP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		GICP	0.0	1.3	1.3	0.7	1.0	0.3	0.7	1.0	1.7
		Robust ICP	0.3	0.0	0.0	0.0	0.3	0.3	1.0	0.3	0.3
		CPD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		IMLP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5A-iii	20%	ICP	0.3	0.0	0.0	0.3	0.3	0.0	0.0	0.0	0.3
		GICP	5.7	7.0	6.3	5.7	8.0	9.3	5.3	4.3	8.3
		Robust ICP	0.3	0.3	0.3	0.3	1.0	0.3	0.0	0.0	0.0
		CPD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		IMLP	0.0	0.3	0.0	1.0	0.3	1.0	1.7	0.7	0.0
5A-iv	30%	ICP	1.3	0.3	1.0	0.7	0.3	1.0	0.3	1.0	0.3
		GICP	17.0	13.3	18.7	15.7	21.3	15.3	14.3	18.0	15.7
		Robust ICP	4.7	2.0	3.3	2.3	3.0	4.7	1.0	1.3	1.7
		CPD	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.0
		IMLP	4.7	3.0	5.3	3.3	6.0	3.3	3.3	3.0	4.0
5B-i	5%	ICP	0.3	1.3	0.7	0.3	0.3	0.3	1.7	0.0	0.7
		GICP	1.3	3.0	1.0	1.7	1.0	2.0	1.7	1.0	1.0
		Robust ICP	14.3	9.7	8.3	12.7	15.0	11.3	11.0	12.7	8.0
		CPD	1.0	0.3	0.7	1.7	1.7	1.0	0.7	1.3	1.7
		IMLP	1.0	0.3	0.3	0.3	0.0	0.7	2.0	0.7	1.3
5B-ii	10%	ICP	1.0	0.3	0.3	1.7	1.0	0.7	0.3	0.7	0.7
		GICP	1.7	2.7	2.3	3.0	3.3	3.3	2.0	2.7	2.3
		Robust ICP	12.7	12.3	11.3	11.0	12.0	10.7	10.7	11.3	9.3
		CPD	1.0	1.7	1.7	0.7	1.0	1.7	1.3	0.7	0.7
		IMLP	1.0	0.3	1.3	2.0	0.7	1.0	1.3	0.7	1.0
5B-iii	20%	ICP	0.7	0.7	1.0	1.3	0.7	0.0	0.3	0.7	0.7
		GICP	5.0	5.0	8.0	9.0	9.7	8.3	8.7	5.3	7.3
		Robust ICP	10.0	7.0	9.7	13.7	11.0	14.7	12.7	10.0	11.3
		CPD	1.0	2.3	1.3	1.3	1.3	2.3	0.7	1.7	1.0
		IMLP	2.7	4.3	2.0	4.7	6.7	4.7	5.7	3.7	3.3
5B-iv	30%	ICP	2.7	1.0	1.7	1.3	1.3	1.3	2.0	2.0	0.7
		GICP	15.0	17.7	12.3	17.7	19.3	20.7	18.3	21.7	20.7
		Robust ICP	16.3	10.0	15.3	15.3	16.0	15.3	12.7	17.0	16.0
		CPD	2.0	3.3	3.3	3.7	1.7	2.0	2.0	1.7	2.3
		IMLP	18.7	12.0	16.0	16.7	19.0	20.7	15.3	16.7	15.7

Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees) in Experiment 5A and $[30, 60]$ mm (degrees) in Experiment 5B, and registered back to a point-cloud representation of the model. The test cases represent the different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shapes constituting 5% (-i), 10% (-ii), 20% (-iii), and 30% (-iv) of the data points. For each test case, 300 randomized trials were conducted. This table reports the percent of unsuccessful registrations (TRE > 10 mm) for each algorithm and test condition.

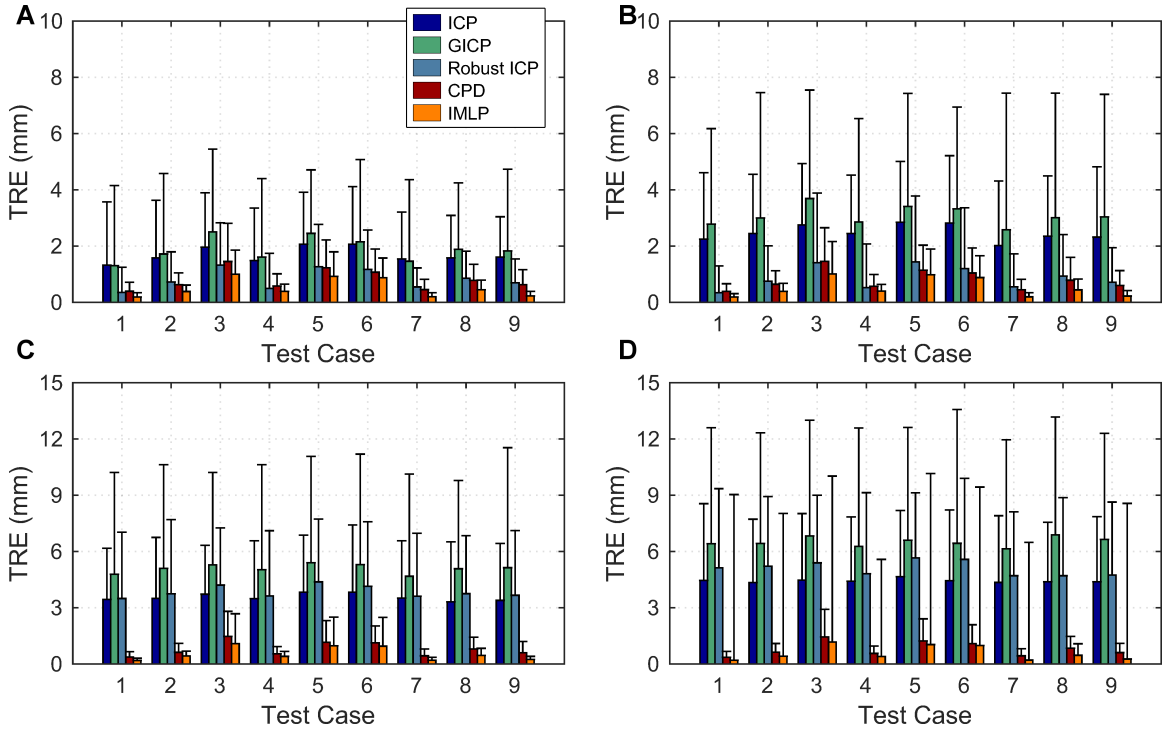

Figure 4.23: Experiment 5A: the median and 95th percentile order statistics of the TRE outcomes for registering a data shape containing outliers to a point-cloud representation of a model shape under moderate misalignment Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees), and registered back to a point-cloud representation of the mesh. The test cases represent different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting (A) 5%, (B) 10%, (C) 20%, and (D) 30% of the data points. For each test case, 300 randomized trials were conducted. The bar graphs and error bars show, respectively, the median and 95th percentile order statistics of the TRE outcomes for each test case. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

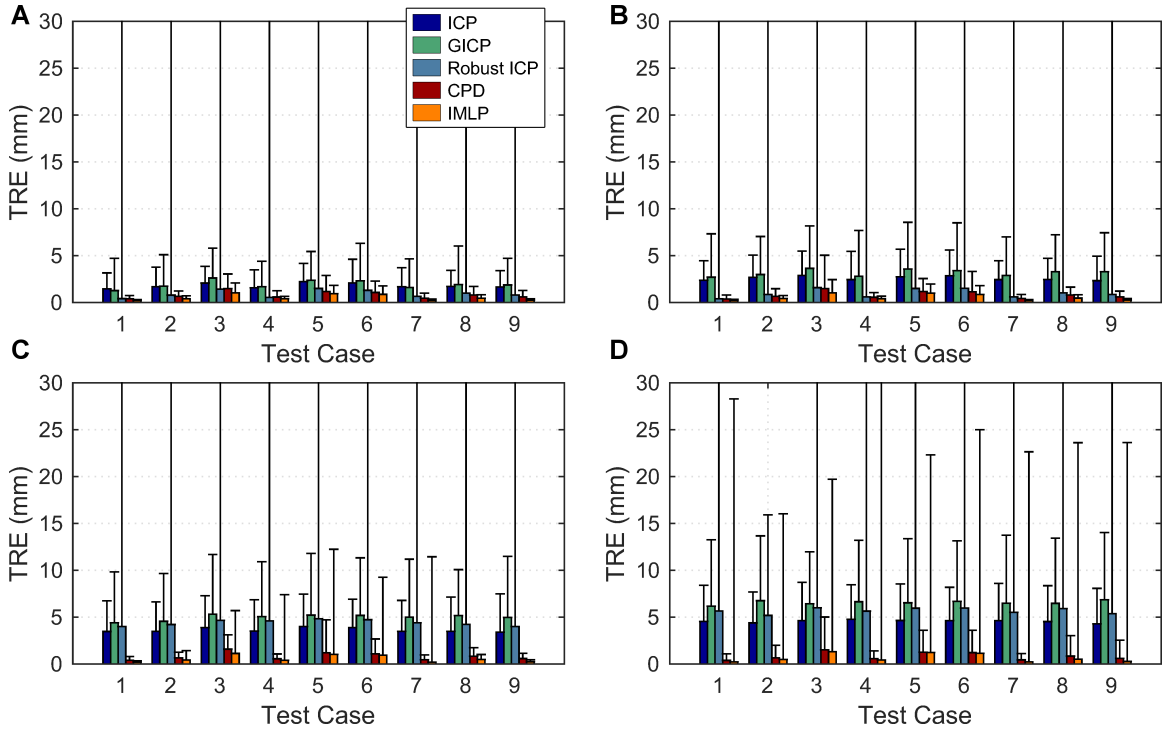

Figure 4.24: Experiment 5B: the median and 95th percentile order statistics of the TRE outcomes for registering a data shape containing outliers to a point-cloud representation of a model shape under large misalignment Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[30, 60]$ mm (degrees), and registered back to a point-cloud representation of the mesh. The test cases represent different noise models used to generate data-shape noise (Table 4.4). Outliers were added to the data shape constituting (A) 5%, (B) 10%, (C) 20%, and (D) 30% of the data points. For each test case, 300 randomized trials were conducted. The bar graphs and error bars show, respectively, the median and 95th percentile order statistics of the TRE outcomes for each test case. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

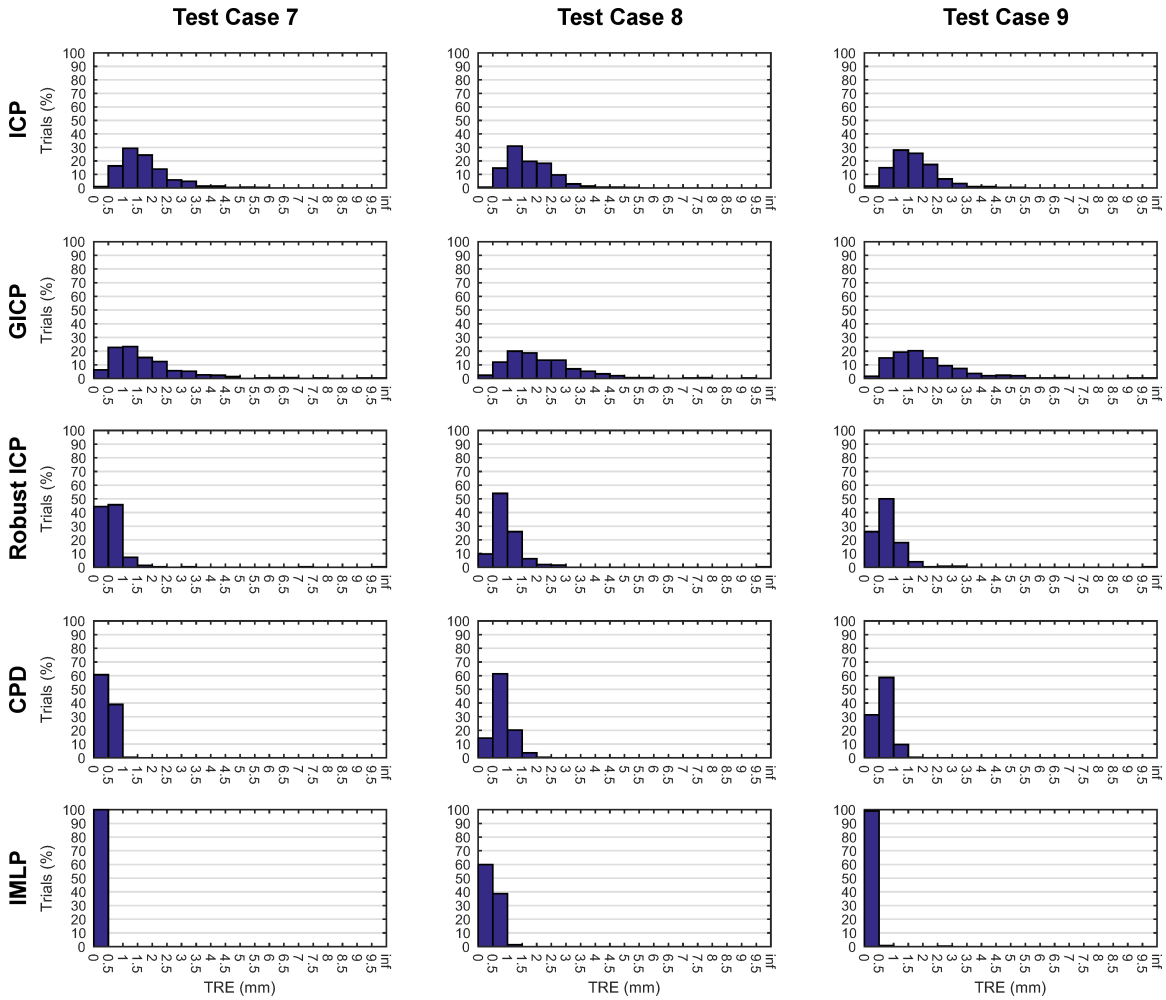

Figure 4.25: Experiment 5A-i: histograms of the TRE outcomes for registering a data shape containing 5% outliers to a point-cloud representation of a model shape under moderate misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees), and registered back to a point-cloud representation of the mesh. Test cases 7–9 represent a subset of the different noise models used to generate noise in the data shape (Table 4.4). Outliers were also added to the data shape constituting 5% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

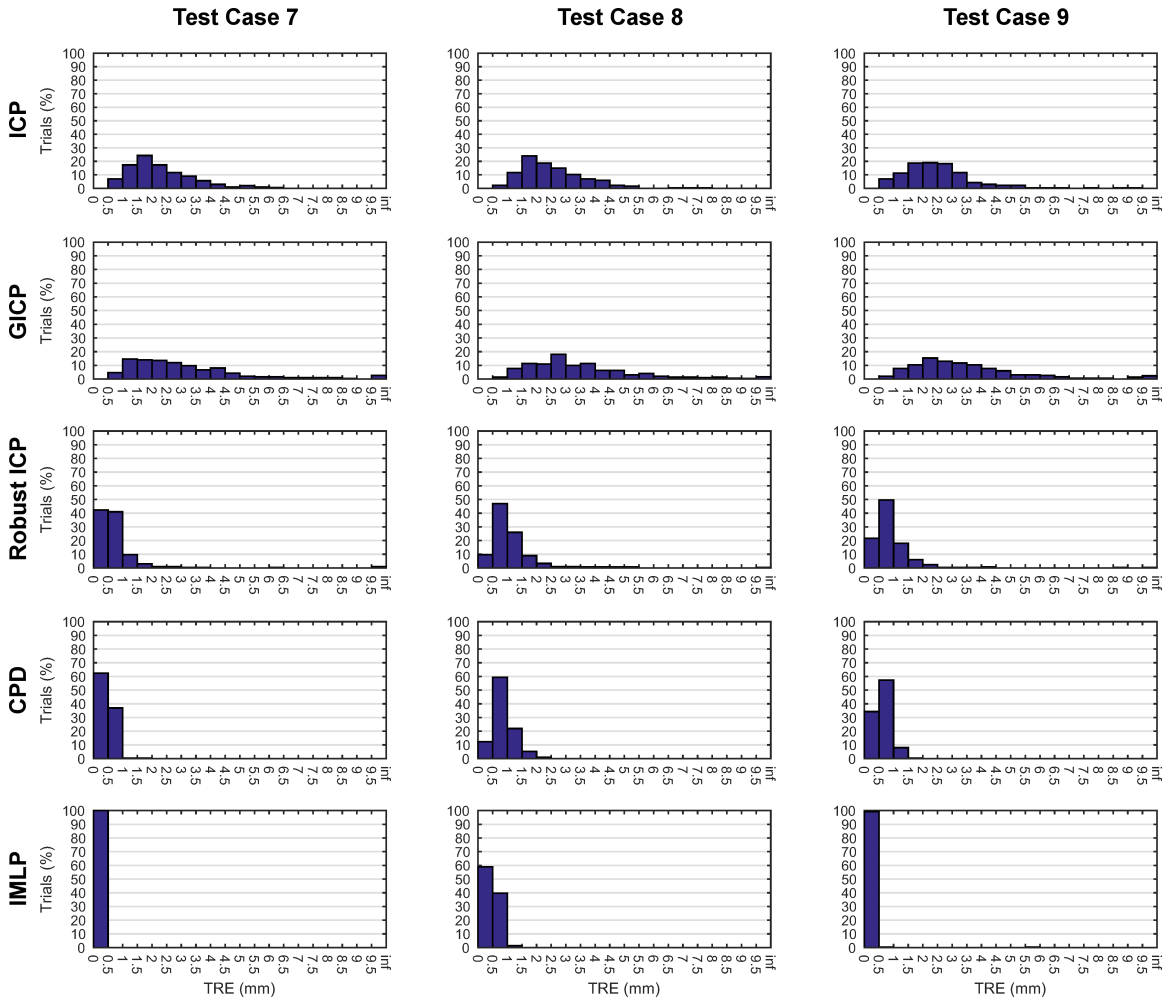

Figure 4.26: Experiment 5A-ii: histograms of the TRE outcomes for registering a data shape containing 10% outliers to a point-cloud representation of a model shape under moderate misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees), and registered back to a point-cloud representation of the mesh. Test cases 7–9 represent a subset of the different noise models used to generate noise in the data shape (Table 4.4). Outliers were also added to the data shape constituting 10% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

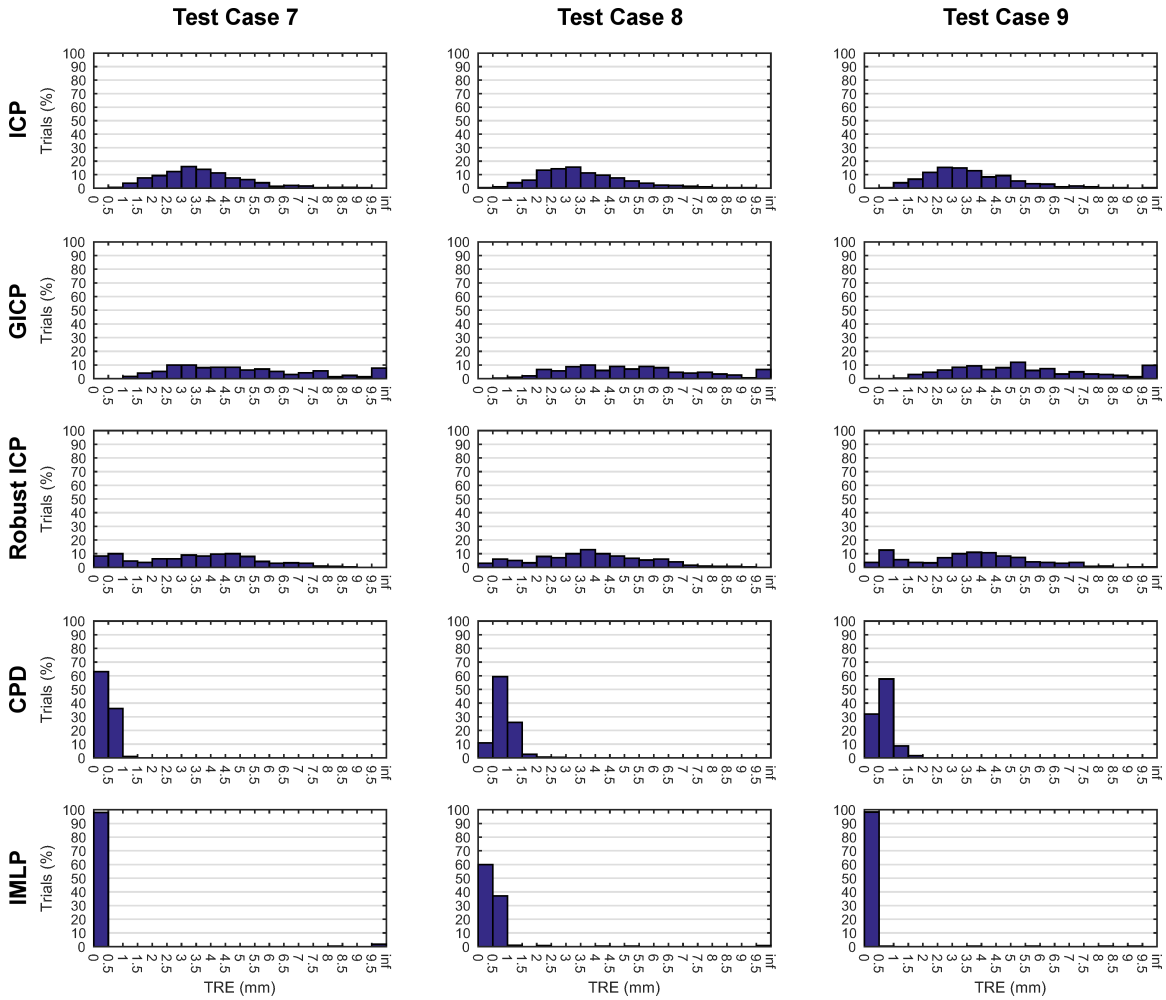

Figure 4.27: Experiment 5A-iii: histograms of the TRE outcomes for registering a data shape containing 20% outliers to a point-cloud representation of a model shape under moderate misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees), and registered back to a point-cloud representation of the mesh. Test cases 7–9 represent a subset of the different noise models used to generate noise in the data shape (Table 4.4). Outliers were also added to the data shape constituting 20% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

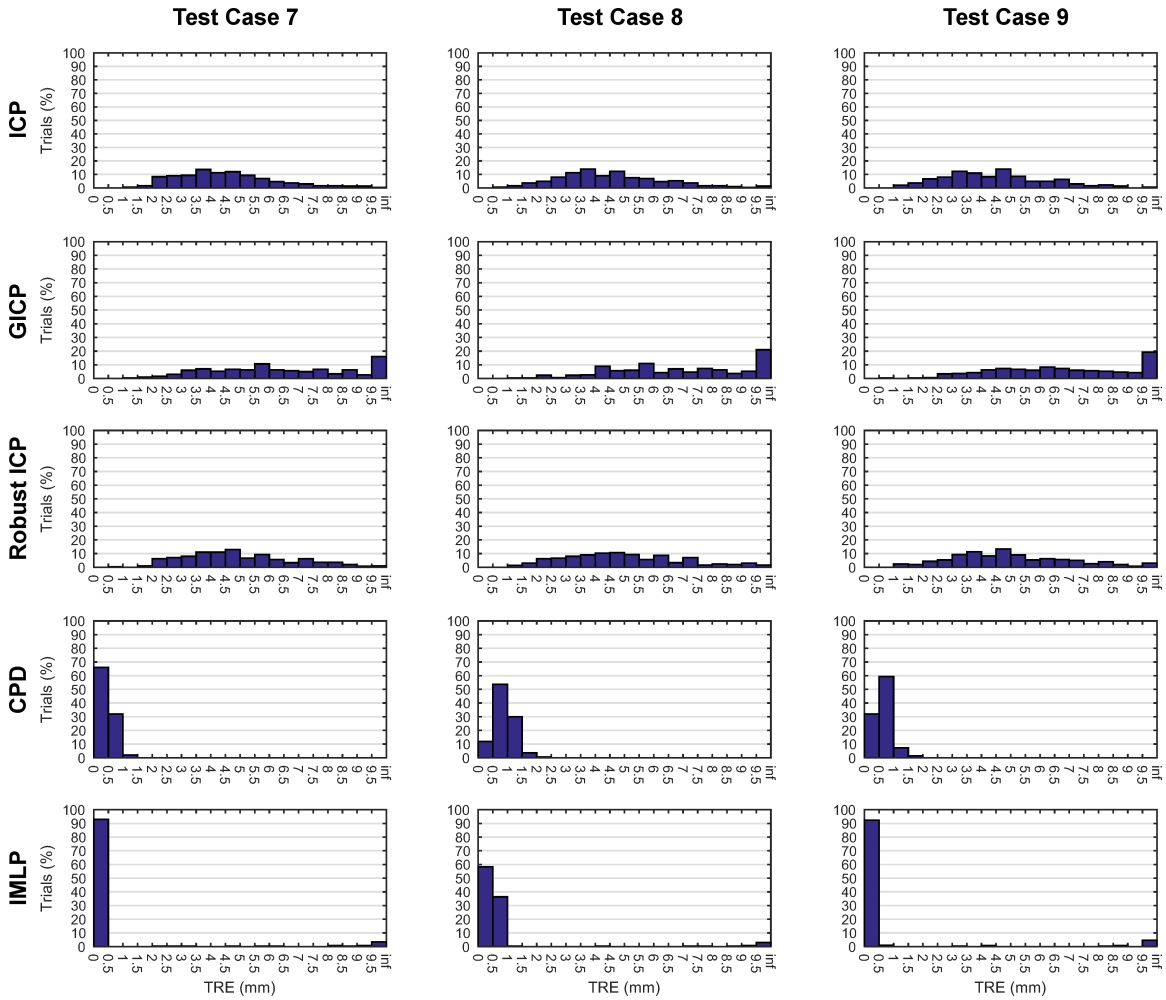

Figure 4.28: Experiment 5A-iv: histograms of the TRE outcomes for registering a data shape containing 30% outliers to a point-cloud representation of a model shape under moderate misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[15, 30]$ mm (degrees), and registered back to a point-cloud representation of the mesh. Test cases 7–9 represent a subset of the different noise models used to generate noise in the data shape (Table 4.4). Outliers were also added to the data shape constituting 30% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

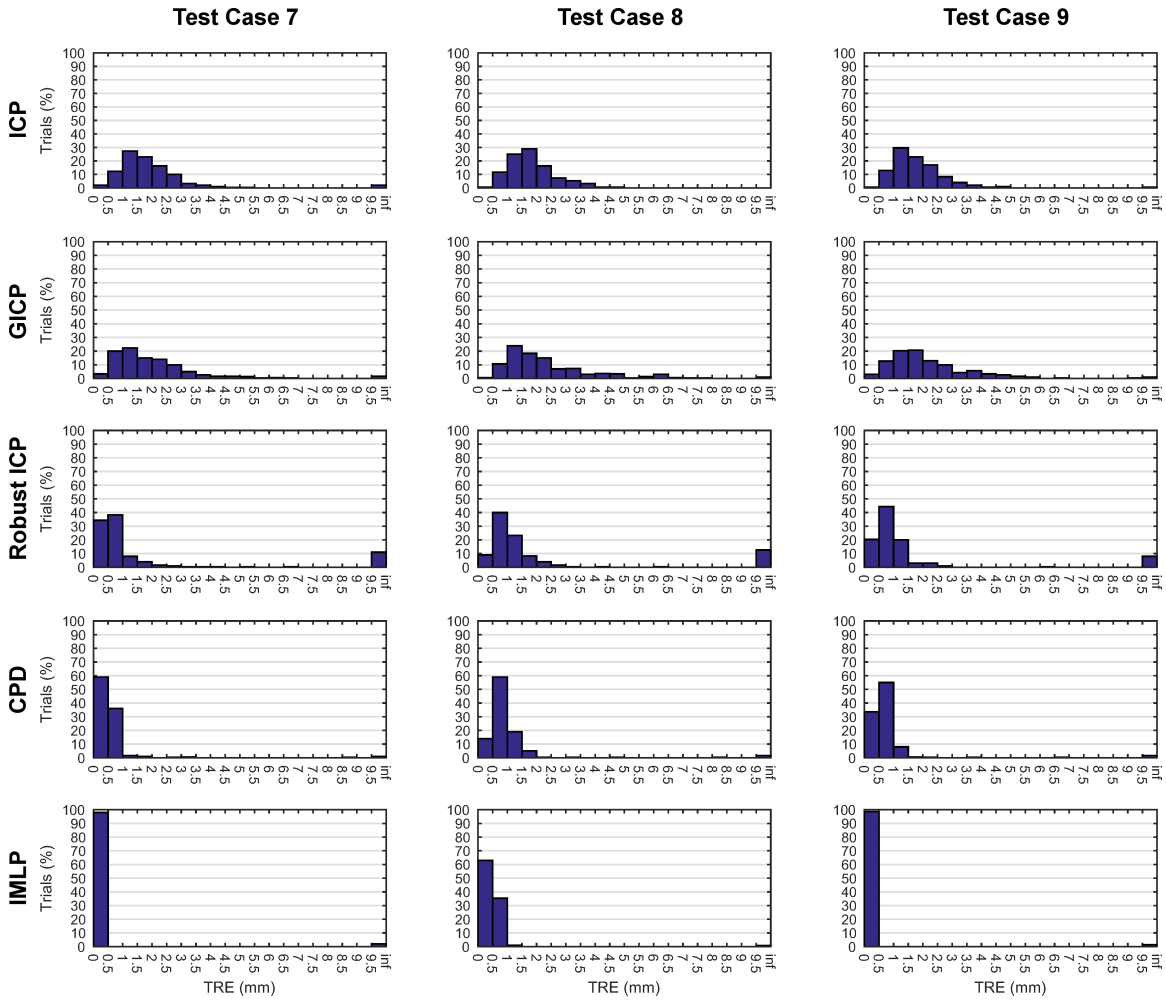

Figure 4.29: Experiment 5B-i: histograms of the TRE outcomes for registering a data shape containing 5% outliers to a point-cloud representation of a model shape under large misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[30, 60]$ mm (degrees), and registered back to a point-cloud representation of the mesh. Test cases 7–9 represent a subset of the different noise models used to generate noise in the data shape (Table 4.4). Outliers were also added to the data shape constituting 5% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

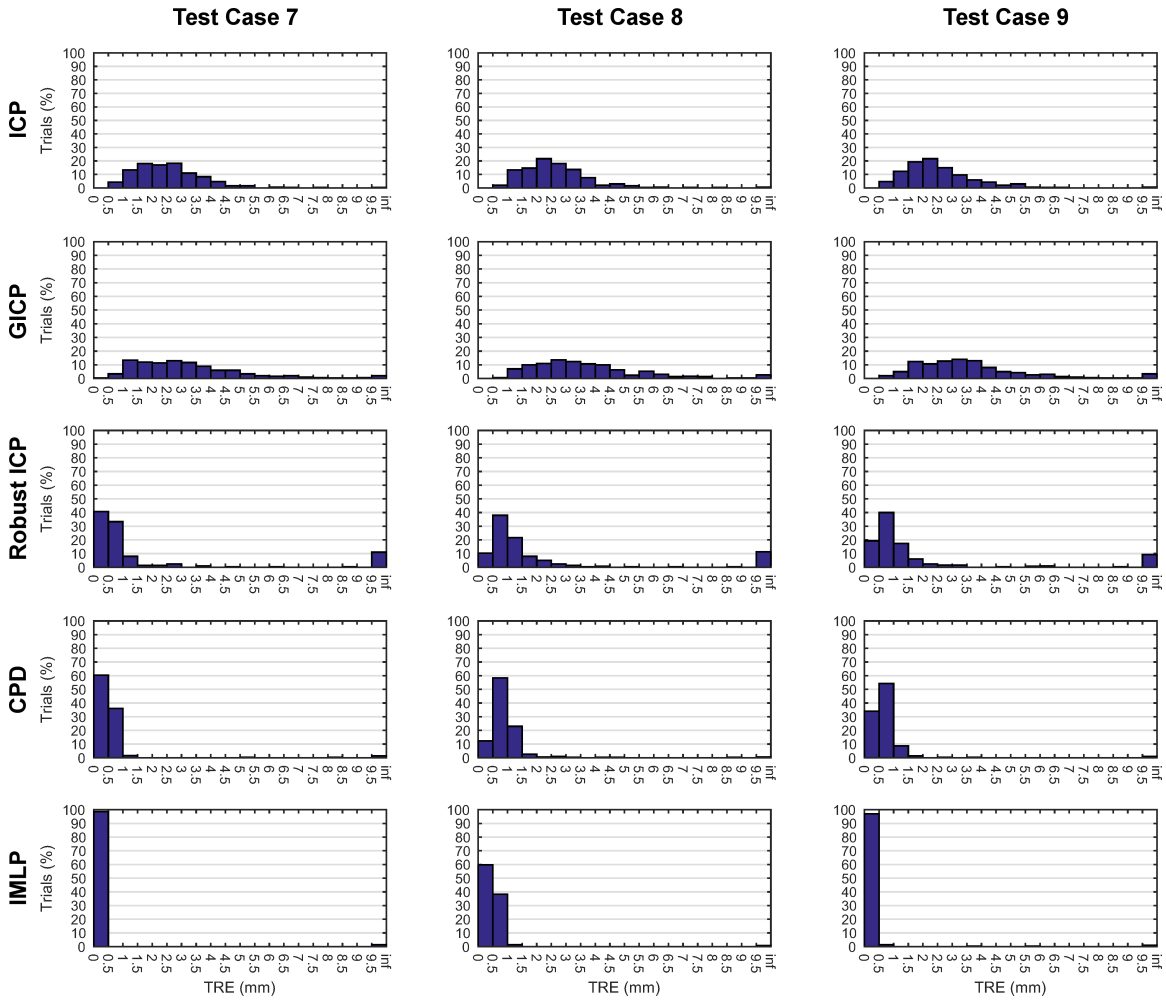

Figure 4.30: Experiment 5B-ii: histograms of the TRE outcomes for registering a data shape containing 10% outliers to a point-cloud representation of a model shape under large misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[30, 60]$ mm (degrees), and registered back to a point-cloud representation of the mesh. Test cases 7–9 represent a subset of the different noise models used to generate noise in the data shape (Table 4.4). Outliers were also added to the data shape constituting 10% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

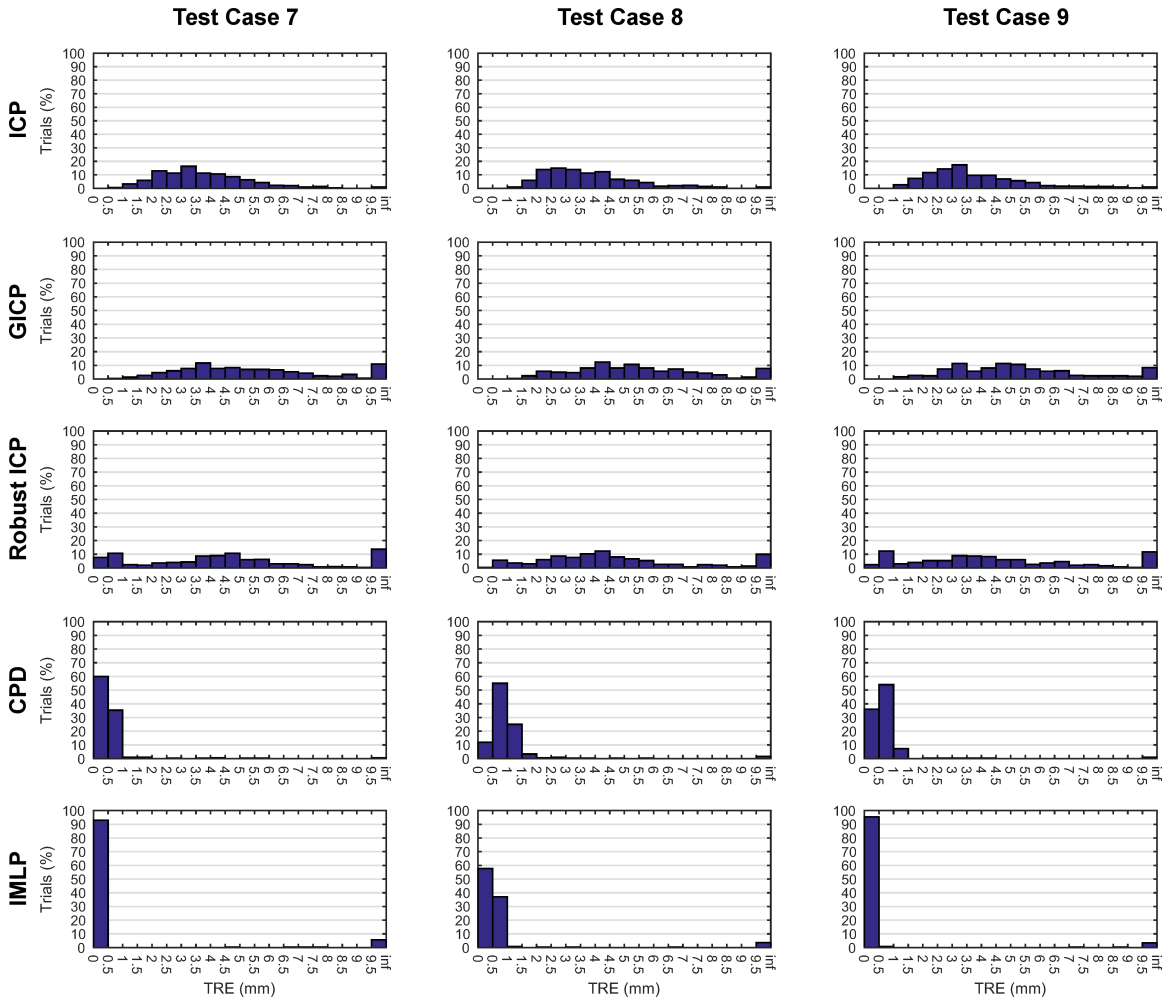

Figure 4.31: Experiment 5B-iii: histograms of the TRE outcomes for registering a data shape containing 20% outliers to a point-cloud representation of a model shape under large misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[30, 60]$ mm (degrees), and registered back to a point-cloud representation of the mesh. Test cases 7–9 represent a subset of the different noise models used to generate noise in the data shape (Table 4.4). Outliers were also added to the data shape constituting 20% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

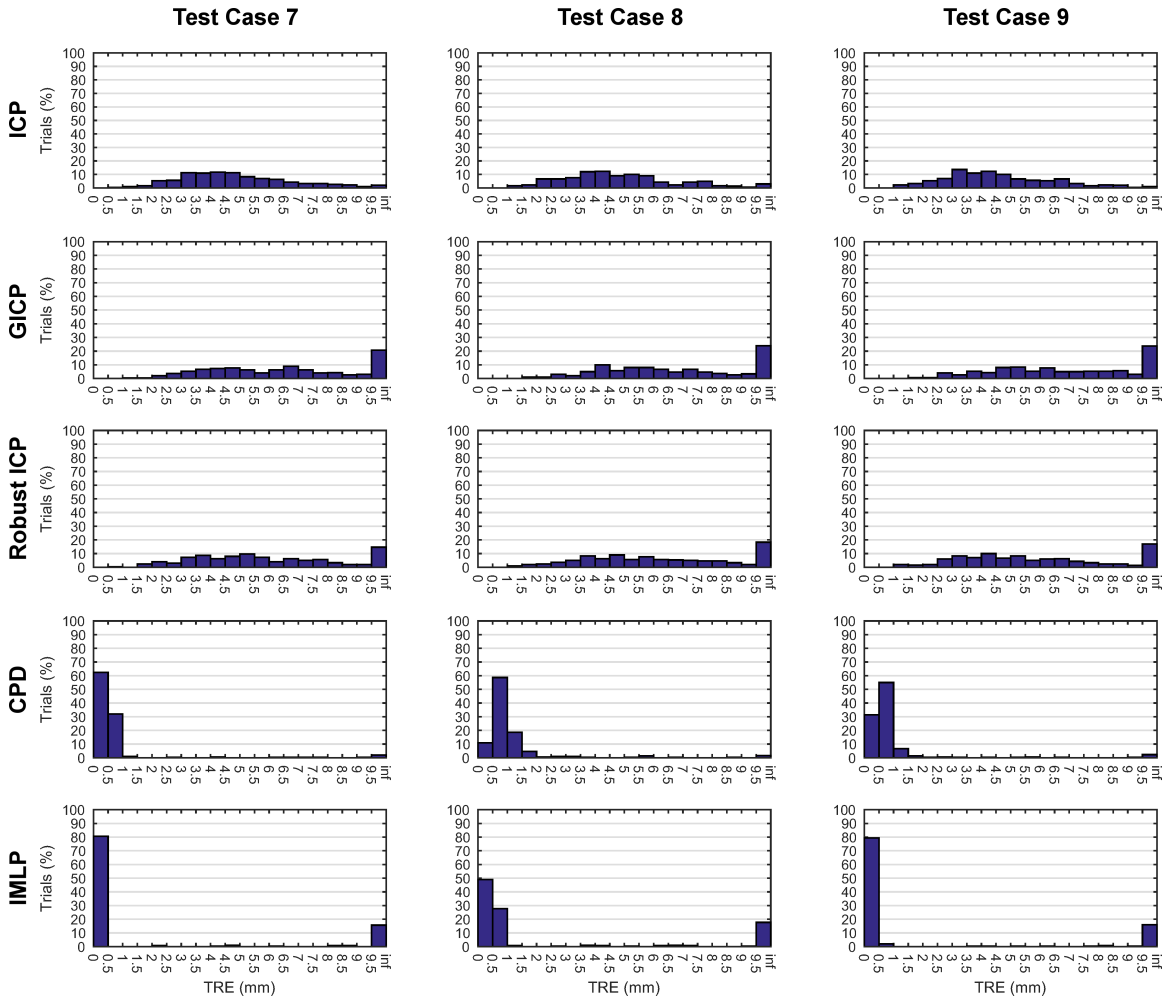

Figure 4.32: Experiment 5B-iv: histograms of the TRE outcomes for registering a data shape containing 30% outliers to a point-cloud representation of a model shape under large misalignment. Data shapes were randomly generated from a mesh model of a human pelvis (Figure 4.2A), misaligned by $[30, 60]$ mm (degrees), and registered back to a point-cloud representation of the mesh. Test cases 7–9 represent a subset of the different noise models used to generate noise in the data shape (Table 4.4). Outliers were also added to the data shape constituting 30% of the data points. For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] a robust variant of ICP,[37] and CPD.[46]

4.5.6 Experiment 6: Sub-Shape Registration

This study investigates the challenging problem of registering sub-shape data to a complete shape model, which arises when data-shape measurements are taken from a sub-region of a shape to be registered. This scenario is investigated by simulating random measurements from a sub-region of a proximal human femur that has been segmented from CT imaging to form a mesh of the bone surface (Figure 4.2B).

The experimental procedure for this study parallels that of Experiment 4 (Section 4.5.4), except that the area sampled for the data shape is confined to a sub-region of the model shape, which is represented by the darkly shaded region of the mesh in Figure 4.2B. Random misalignments for this study are generated on the interval $[10, 20]$ mm translation and $[10, 20]$ degrees rotation. The same algorithms as in Experiment 4 are evaluated using identical settings except that the maximum iteration count for CPD was increased to 200, as 100 iterations was insufficient (other algorithms did not require this increase).

In order to diversify the range of noise conditions considered overall, a different set of noise conditions was investigated for this study. Table 4.11 defines the seven noise models investigated, which includes the zero-noise case, two magnitudes of isotropic noise, two cases of surface-oriented noise to test high variance in both surface-normal and surface-parallel directions, and two test cases involving randomly oriented noise models applied at a global and at a per-point scale. For the test cases involving randomly oriented noise models, different covariances were randomly generated for

Table 4.11: Generative noise models (test cases) used in the randomized registration trials of Experiment 6.

	Test Cases						
Covariance	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Orientation	-	-	-	Surface	Surface	Random-Global	Random-Per-Point
$\lambda_1^{1/2}$	0.0	0.5	1.0	1.0	0.5	0.5	0.5
$\lambda_{2,3}^{1/2}$	0.0	0.5	1.0	0.5	1.0	1.0	1.0

This table defines the covariances used to generate the zero-mean, multi-variate, Gaussian noise applied to the data shape in each test case of Experiment 6. The table defines both the eigenvalues (λ_1, λ_2, λ_3) (magnitude) and the eigenvectors (orientation) of the covariance matrices for each test case. Orientation "Surface" defines the eigenvectors relative to the orientation of the surface at each data-shape point, where λ_1 is the variance in the surface-normal direction and (λ_2, λ_3) are the variances in the surface-parallel directions. Orientation "Random-Global" defines the matrix of eigenvectors to be a randomly generated rotation matrix, which is applied globally over the data shape (i.e. every point in the data shape has the same noise model). Orientation "Random-Per-Point" is similar to the "Random-Global" case except that every point in the data shape is associated with a different randomly generated rotation (i.e. every point in the data shape has a different noise model). Orientation "-" means that the noise model is isotropic and thus unaffected by the choice of eigenvectors.

each trial. As in the prior experiments, 300 randomized trials were conducted for each test case.

Results from Experiment 6 are now discussed. The average TREs of the successful trials (trials with TRE < 10 mm) and the registration failure rates for each algorithm and test case are reported in Figure 4.33 and in Table 4.12, respectively. The ICP algorithm has poor performance across the board in terms of both registration error (above 3.5 mm) and failure rate (approximately 11 - 19%). CPD has the best success rate with almost no failures overall. The failure rates of other anisotropic methods are approximately on-par with the IMLP failure rate of 3 - 7%, with the IMLP-CP variant being marginally lower than the others. As seen in the figure, IMLP achieves

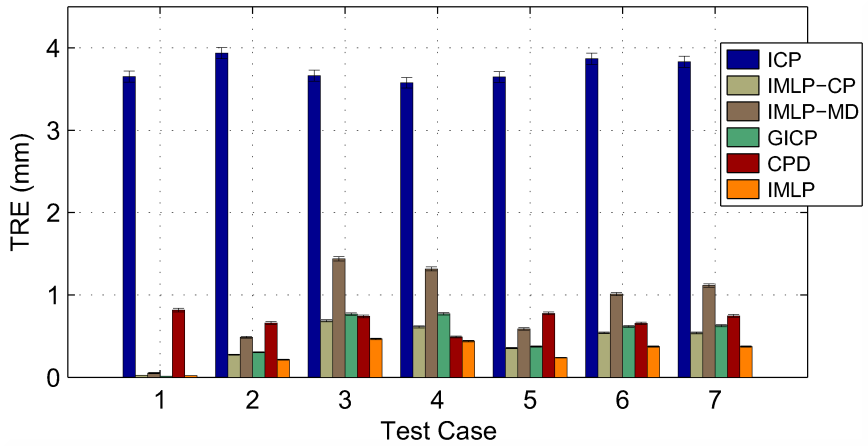

Figure 4.33: Experiment 6: average TREs of the successful registration trials (trials with TRE $< 10\,\text{mm}$) for registering a data shape without outliers to a point-cloud representation of a model shape, where the data shape represents a sub-region of the model shape. Data shapes were randomly generated from a sub-region of a mesh model of a human proximal femur (Figure 4.2B), misaligned by $[10, 20]\,\text{mm}$ (degrees), and registered back to a point-cloud representation of the model. The test cases represent the different noise models used to generate data-shape noise (Table 4.11). For each test case, 300 randomized trials were conducted, with successful registrations being used to compute an average TRE. The error bars show approximate standard deviations of the reported average TRE values. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] and CPD,[46] as well as relative to near comparisons of GTLS-ICP[55] and A-ICP[57] using the two variants IMLP-CP and IMLP-MD, respectively, which modify IMLP's most-likely-match criterion to that of closest-point and Mahalanobis-distance matching.

the lowest registration error in every test case. Compared to CPD the improvement in registration error by IMLP is often substantial, especially for the zero-noise case where IMLP achieves nearly zero registration error and CPD has an average TRE near 1 mm. Again, we find that the Mahalanobis-distance-matching criterion, as assessed by IMLP-MD, computes substantially worse registration error than the closest-point-matching criterion used by IMLP-CP and also by GICP; on the other hand, the most-likely-point-matching criterion of IMLP achieves the lowest registration error in every case.

Table 4.12: Experiment 6: registration failure rates for registering a sub-shape without outliers.

	Failure Rate (%) by Test Case						
Alg	**1**	**2**	**3**	**4**	**5**	**6**	**7**
ICP	15.0	10.7	17.3	13.7	14.7	18.7	16.3
IMLP-CP	4.7	2	5.3	4.3	4.3	4.3	4.0
IMLP-MD	6.0	3.3	7.3	5.3	7.0	6.7	5.3
GICP	6.0	4.3	8.3	6.3	6.0	5.3	4.7
CPD	0.0	0.0	0.0	0.0	0.0	0.3	0.3
IMLP	6.0	3.0	7.0	5.0	6.0	6.3	5.0

Data shapes were randomly generated from a mesh model of a human femur (Figure 4.2B), misaligned by $[10, 20]$ mm (degrees), and registered back to a point-cloud representation of the model. The test cases represent different noise models used to generate data-shape noise (Table 4.11). For each test case, 300 randomized trials were conducted. This table reports the percent of unsuccessful registrations (TRE > 10 mm) for each algorithm and test case.

Order statistics for the TRE outcomes, including the median and 95th percentile values, are shown in Figure 4.34. The order statistic outcomes roughly parallel those of the average TRE outcomes as described for Figure 4.33, with one notable difference being that the median TRE for CPD is substantially better than the average TRE in the first test case. Finally, histograms of the TRE outcomes from the registration trials for test cases 7–9 (Table 4.4) are shown in Figure 4.35, which show the distribution of the TRE outcomes in greater detail.

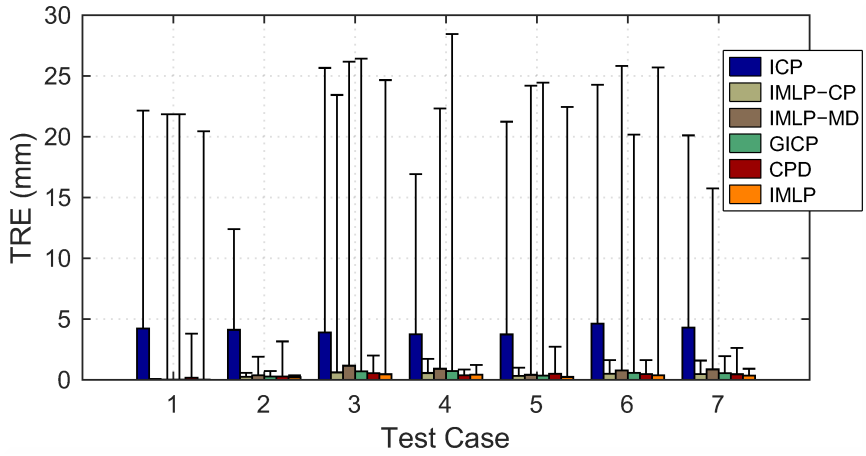

Figure 4.34: Experiment 6: the median and 95th percentile order statistics of the TRE outcomes for registering a data shape without outliers to a point-cloud representation of a model shape, where the data shape represents a sub-region of the model shape. Data shapes were randomly generated from a sub-region of a mesh model of a human proximal femur (Figure 4.2B), misaligned by $[10, 20]$ mm (degrees), and registered back to a point-cloud representation of the model. The test cases represent the different noise models used to generate data-shape noise (Table 4.11). For each test case, 300 randomized trials were conducted. The bar graphs and error bars show, respectively, the median and 95th percentile order statistics of the TRE outcomes for each test case. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] and CPD,[46] as well as relative to near comparisons of GTLS-ICP[55] and A-ICP[57] using the two variants IMLP-CP and IMLP-MD, respectively, which modify IMLP's most-likely-match criterion to that of closest-point and Mahalanobis-distance matching.

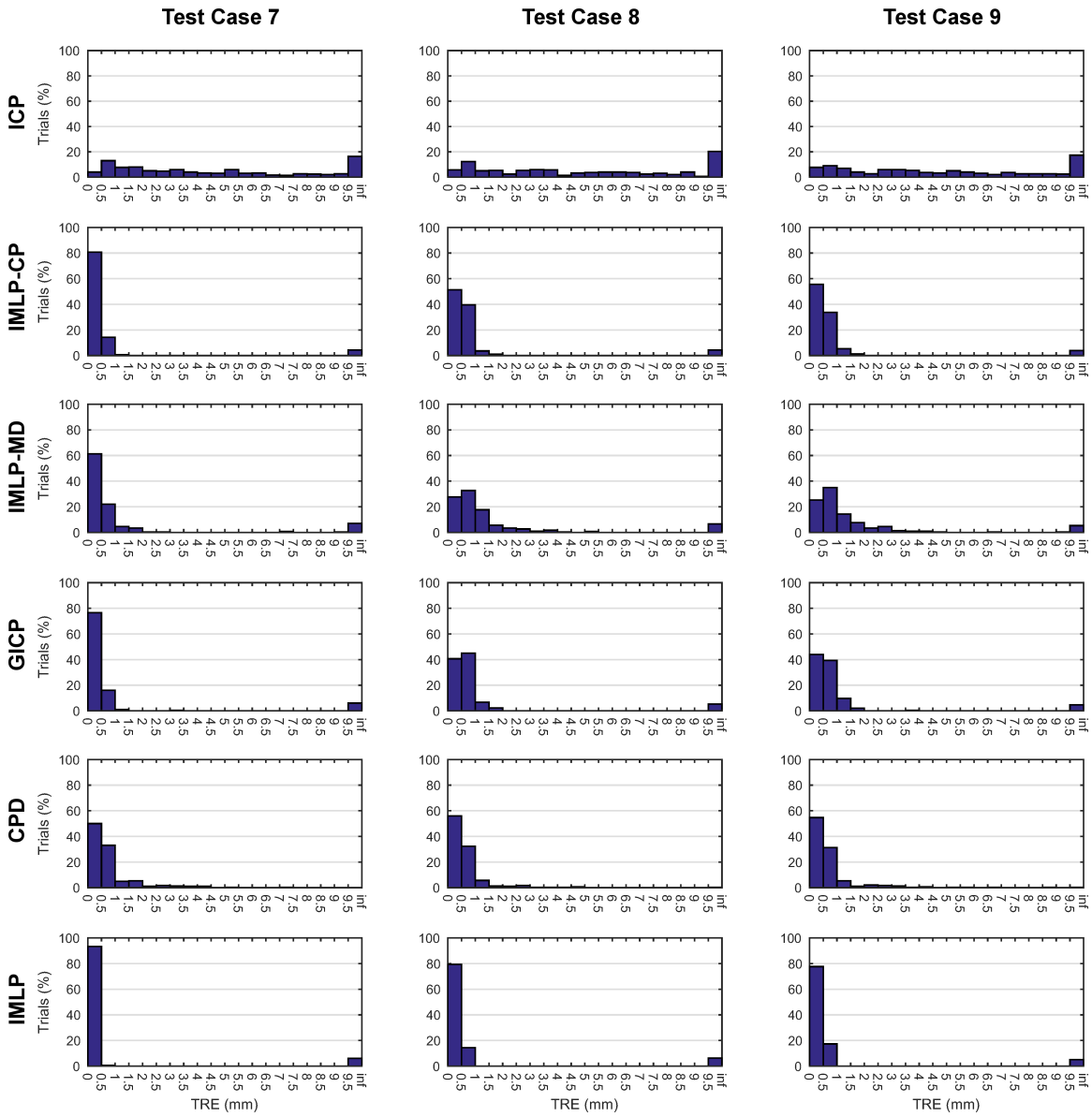

Figure 4.35: Experiment 6: histograms of the TRE outcomes for registering a data shape without outliers to a point-cloud representation of a model shape, where the data shape represents a sub-region of the model shape. Data shapes were randomly generated from a sub-region of a mesh model of a human proximal femur (Figure 4.2B), misaligned by $[10, 20]$ mm (degrees), and registered back to a point-cloud representation of the model. Test cases 7–9 represent a subset of the different noise models used to generate data-shape noise (Table 4.11). For each test case, 300 randomized trials were conducted. The proposed IMLP algorithm was evaluated relative to ICP,[4] GICP,[56] and CPD,[46] as well as relative to near comparisons of GTLS-ICP[55] and A-ICP[57] using the two variants IMLP-CP and IMLP-MD, respectively, which modify IMLP's most-likely-match criterion to that of closest-point and Mahalanobis-distance matching.

4.5.7 Experiment 7: Registering Shapes of Partial Overlap

This study investigates a yet more challenging problem of registering two shapes that have only partial overlap, meaning there are regions on both shapes that do not have a true correspondence with the other shape. To investigate this scenario, a model of the statue Laurana (Figure 4.36A) is used, which was downloaded from the Institute of Science and Technologies (ISTI-CNR), Pisa, Italy, under a Creative Commons License (http://vcg.isti.cnr.it/downloads/3dgallery/form_laurana.htm). A decimation was applied to the original mesh using the Quadric Edge Collapse Decimation feature of MeshLab,[74] reducing the model to 50,000 triangles. The coordinate system was also adjusted in order to position the origin at the mesh centroid. Two divisions of the mesh were then performed to extract the front (Figure 4.36B) and right (Figure 4.36C) half-sections of the model. A dense point cloud for the data shape was formed from the vertices of the right half-section, and a dense point cloud for the model shape was formed from the center points of the triangles of the front half-section. Thus, the region of overlap between the data and model shapes comprised 50% of each shape (Figure 4.36D).

Ten randomized registration trials were performed by applying 0.5 mm standard deviation of isotropic Gaussian noise to the data-shape points and applying misalignments of 10 mm and 10 degrees in random directions. Validation points for computing

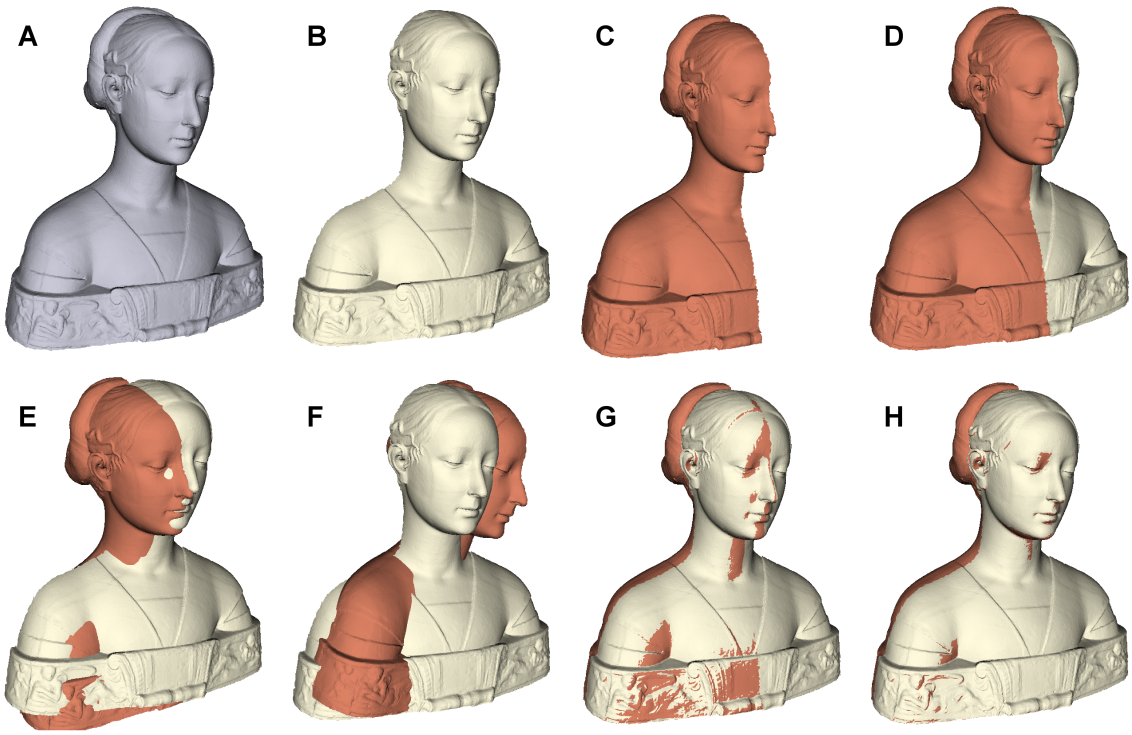

Figure 4.36: Experiment 7: registering shapes of partial overlap. (A) the statue Laurana sub-divided into (B) front and (C) right half-sections, such that (D) a 50% overlap exists between the two sub-shapes. The sub-shapes were (E) misaligned by 10 mm and 10 degrees in a random direction and then registered using (F) CPD,[46] (G) GICP,[56] and (H) the proposed IMLP algorithm. Sub-figures (E-H) show the initial misalignment and the final registered alignments of the two shapes for the 6th randomized trial of Experiment 7.

the TRE were selected randomly from the model shape, not being confined to the region of overlap.

The algorithms evaluated in this study were GICP,[56] CPD,[46] and IMLP. Various values were experimentally tested for the match threshold distance of GICP, with 3 mm finally selected as having the lowest TRE. For CPD, various outlier weights were tested with poor results obtained in every case; we therefore applied the standard outlier weight of 0.5. For IMLP, we used the default chi-square inverse CDF threshold (χ^2_{thresh}) value of 7.81. To minimize bias from the non-overlapping region, matches identified as outliers were configured to be completely disregarded in the registration phase of the IMLP algorithm (rather than inflating their variances), as suggested in Section 4.2 for this type of application. The measurement-error covariances of IMLP were set to zero in this study in order to, as much as possible, enable the noise model to adapt to the region of overlap based on the match uncertainty term. To restrict "good" matches to the region of overlap, the max threshold for the match uncertainty parameter (σ^2_{max}) was set to 0.1 mm^2. The surface-model covariances for both GICP and IMLP were set to the same values as used in Experiment 4 (Section 4.5.4).

Table 4.13 shows the TRE computed by each algorithm for each of the 10 randomized trials. As indicated in the table, CPD is unable to properly register any of the trials in this scenario, whereas both IMLP and GICP register all of them nearly perfectly, with GICP having a moderate to slight accuracy advantage. The bottom row of Figure 4.36 provides a visualization of the initial shape misalignment (Fig-

Table 4.13: Experiment 7: TRE outcomes for registering shapes of partial overlap.

	TRE (mm) per Trial									
Alg	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
CPD	63.533	69.988	82.186	83.449	72.612	71.378	69.431	79.822	78.071	68.675
GICP	0.138	0.150	0.187	0.060	0.124	0.165	0.272	0.252	0.158	0.242
IMLP	0.264	0.306	0.283	0.224	0.352	0.294	0.295	0.277	0.250	0.365

TRE is reported for each of 10 ramdomized registration trials conducted for Experiment 7, which involve registering two half-sections of the statue Laurana that have 50% true overlap (Figure 4.36D). The half-sections were randomly misaligned by 10 mm and 10 degrees and then registered using CPD,[46] GICP,[56] and the proposed IMLP algorithm.

ure 4.36E) and of the registered alignments computed by each algorithm, including CPD (Figure 4.36F), GICP (Figure 4.36G), and IMLP (Figure 4.36H). The visualizations of Figures 4.36E-H represent the 6th randomized registration trial of this experiment.

4.5.8 Experiment 8: Runtime Comparison of Methods for Computing the Most Likely Matches

As a final study, an investigation is made concerning the speedup afforded by the PD-tree search technique used to compute the most likely matches for IMLP. We compare the runtime of a naive exhaustive search to that of the proposed PD-tree method using both the spherical (4.13) and simple ellipsoidal (4.14) bounding techniques, as described in Sections 4.3.2 and 4.3.3, respectively.

Table 4.14 shows the average runtimes obtained from running IMLP over all 300 trials of test case 1 from Experiment 4B using each method of computing the most

Table 4.14: Experiment 8: IMLP runtime comparison using different PD-tree bounding methods for computing the most likely matches.

Search Method	Naive Search	Spherical Bound	Simple Ellipsoidal Bound
Runtime (sec.)	18.523	0.141	0.128

Average runtimes are reported for the proposed IMLP algorithm over the 300 registration trials of Experiment 4B, test case 1, which involves registering 100 random samples to a point-cloud representation of a pelvis model (Figure 4.2A). Runtimes were recorded for the naive exhaustive search and for the proposed PD-tree search strategy comparing two of the proposed PD-tree bounding methods: the spherical (4.13) and simple ellipsoidal (4.14) bounds. The compact ellipsoidal bound (4.16) was not evaluated.

likely matches. As seen in the table, the proposed PD-tree search strategy achieves more than 140x speedup over the naive search when using the simple ellipsoidal bounding technique. Comparing the two alternative PD-tree bounding methods indicates that the more compact simple ellipsoidal bound achieves approximately 10% greater runtime efficiency than the spherical bound, even though its computations are significantly more complex.

We note that the most compact ellipsoidal bound (4.16), which is not evaluated here, may enable even further speedup over the simple ellipsoidal bound evaluated above. This is a likely outcome, since the runtime computations performed for each ellipsoidal bounding technique are very similar; thus, the most compact bound should provide even better performance.

4.6 Concluding Remarks

A novel variant of the Iterative Closest Point (ICP) algorithm, called the Iterative Most Likely Point (IMLP) algorithm, has been presented that has the ability to compute optimal shape alignment under anisotropic noise conditions by incorporating a probabilistic framework within both the correspondence and registration phases of the algorithm. Another advantage of this framework is the ability to model locally-linear regions of a continuous surface, which greatly improves the registration accuracy that is attainable when using discrete representations of a surface. Dynamic estimation of the match uncertainty enables IMLP to adaptively adjust its noise model to different levels of misalignment, which provides robustness under large initial misalignments and high accuracy and sensitivity to outliers when in the vicinity of the correct solution. In addition, the probabilistic underpinning provides a cohesive and flexible framework for detection and mitigation of outliers, as well as enabling registration of shapes having only partial overlap via a user-defined maximum threshold on the match uncertainty term.

Through an extensive set of experiments, involving more than 50,000 randomized executions of the IMLP algorithm alone, IMLP has been shown to possess significant registration accuracy and robustness advantages compared to long-established and recently introduced algorithms over a broad range of test conditions including various noise conditions, percentages of outliers, ranges of misalignment, and test shapes. Other algorithms evaluated include the long-established algorithms of ICP[4] and the

robust ICP variant by Zhang,[37] as well as the more recent, leading algorithms of GICP[56] and CPD.[46] In addition, close comparison is made to the prior anisotropic registration methods of GTLS-ICP[55] and A-ICP[57] using modifications on our own method, IMLP-CP and IMLP-MD, respectively. Relative to all tested algorithms, IMLP demonstrated a clear accuracy advantage overall.

Compared to CPD, which has a very effective outlier mitigation capability, IMLP was demonstrated to achieve equivalent registration success rates for outlier percentages of 10% and below, with marginal to moderate relative increase in failure rate at 20% outliers and large relative increase at 30% outliers. On the other hand, in terms of the registration accuracy of successful trials, IMLP achieves significantly better or on-par accuracy compared to CPD for all levels of outliers studied (up to 30%). Based on our results, we conclude that IMLP is a very effective method for registering shapes with up to 10% outliers and retains excellent performance at 20% outliers for moderate levels of misalignment.

Only for the experiment involving registration of partially-overlapping shapes did another algorithm (GICP) clearly come ahead of IMLP in terms of registration accuracy. However, IMLP nonetheless demonstrated a strong performance in this scenario and achieved higher accuracy than GICP in all other experiments performed. Further, the CPD algorithm failed completely in this scenario.

A surprising outcome of our experiments reveal that the Mahalanobis-distance-matching criterion consistently performs worse than the closest-point-matching cri-

terion for registrations involving point-cloud model shapes, whereas for the case of a mesh-based model shape the opposite is true. In contrast, the most-likely-point-matching criterion of IMLP provides the best performance in both scenarios. These observations were evaluated using two variants of IMLP—IMLP-CP and IMLP-MD—which incorporate the modified match criteria of closest-point and Mahalnobis-distance matching, respectively, all else being equal.

Although IMLP is several times slower than ICP, it nonetheless provides a very competitive runtime, considering the substantial reduction in registration error that it achieves. Compared to CPD (the next-best performing algorithm overall), IMLP achieves better registration accuracy in the majority of test cases considered, while being approximately two orders of magnitude more efficient. While IMLP is efficient enough to run on its own, further substantial speed-up could be easily obtained by initializing the registration with a faster algorithm such as ICP, as demonstrated in prior work.[57] Furthermore, the computations performed by IMLP are highly parallelizable and may be efficiently implemented on a GPU, as already demonstrated in prior work regarding ICP-based algorithms.[75] Finally, since we have used the simple ellipsoidal bounding method (4.14) for the PD-tree search in our implementation, further speedup may be possible by implementing the more compact ellipsoidal bound of (4.16).

As alluded to in the preceding paragraph, an effective and novel search strategy has been developed for computing the most likely matches on a model shape as defined by

IMLP's most-likely-point-matching criterion. As demonstrated by the experiments, the proposed search strategy provides a massive speedup (>140x) over a naive search. This speedup is a key enabler of the efficient runtime performance achieved by IMLP.

An alternative approach for solving the generalized total-least-squares (GTLS) problem of aligning two corresponding point sets under a generalized noise model was also developed. The proposed approach turns out to be that of a Gauss-Newton-based method, which the experiments demonstrate to be more accurate, efficient, and stable compared to prior solutions proposed for this problem. The proposed approach supports anisotropic error in both sets of points being registered and is easily implemented using a linear least squares solver, which avoids the software dependency of a nonlinear optimization library. In addition to its incorporation within the IMLP algorithm, the GTLS registration approach may also be used to implement related algorithms that incorporate generalized noise models, such as GICP.

4.7 Contributions

The contributions from this chapter include:

2. Iterative Most Likely Point (IMLP) Algorithm[25]

   (a) a robust probabilistic algorithm for registering *positional* feature data that is characterized by *anisotropic* uncertainty; the algorithm dynamically adapts its noise model within each iteration in order to improve conver-

gence towards the correct solution and in order to detect and mitigate outliers

(b) an efficient implementation of the IMLP algorithm consisting of:

i. *IMLP Correspondence Phase*: a fast PD-tree-based search for efficiently computing the most likely matches from a model shape under the assumption of anisotropic Gaussian uncertainty in the data- and/or model-shape positions

ii. *IMLP Registration Phase*: an ad hoc solution to the generalized total least squares problem of registering corresponding point sets that are characterized by anisotropic uncertainty; this solution has the form of a modified Gauss-Newton approach that is more efficient, robust, and accurate than prior methods; further, this new solution is straightforward to implement using a linear least squares solver

4.8 Published Work

Material from this chapter has appeared in the following publication:

1. S. D. Billings, E. M. Boctor, and R. H. Taylor, "Iterative most-likely point registration (IMLP): A robust algorithm for computing optimal shape alignment," *PLoS ONE*, vol. 10, no. 3, p. e0117688, 2015. [Online]. Available: `http://dx.doi.org/10.1371/journal.pone.0117688`

Chapter 5

Iterative Most Likely Oriented Point (IMLOP) Algorithm

This chapter presents the Iterative Most Likely Oriented Point (IMLOP) algorithm, which is an algorithm for registering *oriented feature data* characterized by isotropic uncertainty. Unlike the ICP and IMLP algorithms previously described, the features used by IMLOP include not only position elements, but also orientation elements. A primary example of this form of data is a surface model that includes the surface normal orientations. The IMLOP algorithm is an ICP-based method that incorporates a probabilistic framework to combine the position and orientation information of the shapes being registered. Content of this chapter has appeared in [25] and [27].

There are various methods whereby orientation data (as required by the IMLOP

algorithm) could be measured in the context of clinical applications, such as the clinical applications discussed in Chapter 1. Surface normal estimation by physical probing is well established in the robotics literature and could be applied in surface probing applications, such as THR surgery. A tracked probe equipped with a force/torque sensor (e.g., [76, 77]) or optical waveguide sensing[78] could be used, for example. Alternatively, various range-imaging techniques exist to measure a surface and its normal orientations, e.g., normal estimation from range maps,[6] structured and non-structured light-based sensing,[79] and photometric stereo.[80]

Besides surface-based registration, other forms of oriented data are encountered in clinical applications. Oriented fiducials have been suggested as an alternative to point-based fiducials in order to provide positional information in preoperative images.[28] Blood vessel registration, such as between 3D ultrasound and MRI or CT,[81] is a further example, where vessels may be represented by vessel-centric points having orientations directed along the vessel centerlines.

Prior ICP-based methods making use of orientation data have been proposed. Early methods limited their consideration of orientations to the correspondence phase, where orientation errors were used to filter the choice of matches. Pulli[82] used surface normals to limit the pool of match candidates to points having a normal orientation within 45 degrees of the surface normal associated with the data point being matched. A similar approach was presented by Lara et al.,[83] where allowable bounds on the match orientation errors were adaptively computed. Münch et al.[84] later incorpo-

rated an orientation- and position-based distance metric into both the match and registration phases of an ICP-based method by combining the square Euclidean distance between matched point positions with the square difference of the matched unit normal orientations using a weighted sum. Their implementation employs an approximation for the transformed surface normals that assumes low curvature of the model shape and that requires the model shape to be defined in a differentiable, parametric form. The transformation is treated as a locally affine deformation model that is also based on the model-surface parameterization and solved using the cost function gradient through iterative coordinate descent applied sequentially to each pair of matched features in turn. Kang et al.,[22] developed a 2D–3D method for registration of contours in the context of single X-ray to CT registration, where 2D normal orientations were optimized alongside positions using the von Mises and Gaussian distributions as noise models. This algorithm was formulated within an expectation maximization (EM) framework and solved using particle swarm optimization. Another recent method for 2D–3D registration of oriented data was proposed by Baka et al.,[23] which used Gaussian distributions to model both orientation and position data within a Gaussian mixture model (GMM) framework for registering blood vessels in X-ray and CT angiography.

The primary similarities and differences between the IMLOP algorithm developed in this chapter and those of the most relevant prior works are now summarized. The probabilistic framework of IMLOP is most similar to that of Kang et al.[22] Un-

like IMLOP, however, the method by Kang et al. is a 2D–3D (rather than 3D–3D) registration method that uses 2D noise models within a different type of registration framework based on EM. In contrast to the algorithms of Pulli[82] and Lara et al.,[83] which use orientations to narrow the range of permitted matches, the IMLOP algorithm directly combines the orientation and position data within a cohesive probabilistic framework by optimizing with respect to a position- and orientation-based match likelihood function within both the correspondence and registration phases of the algorithm. It is this characteristic that motivates the "most likely oriented point" name of this algorithm. Finally, unlike the algorithm by Münch et al.,[84] the IMLOP algorithm does not require any special parameterization of the model shape and uses a distance metric motivated by a probabilistic interpretation of the registration problem.

5.1 Probabilistic Model

The IMLOP algorithm incorporates a probabilistic framework formulated using isotropic Gaussian and Fisher distributions to model the measurement errors of the position and orientation data, respectively. Since the Fisher distribution is the analog of the Gaussian distribution on the unit sphere,[85] pairing these distributions to model oriented-point measurement error is both natural and analytically convenient (see Appendix D for a description and example illustrations of the Fisher distribution).

Assuming zero-mean, independent, identically distributed (iid) error for both the orientation- and position-based data, the *match likelihood function* for an oriented data point, $\mathbf{x} = (\mathbf{x}_\mathrm{p}, \hat{\mathbf{x}}_\mathrm{n})$, that has been transformed by a current registration estimate, $[\mathbf{R},\, \mathbf{t}]$, is defined as

$$f_\mathrm{match}(\mathbf{x} \,|\, \mathbf{y}, \sigma^2, \kappa, \mathbf{R}, \mathbf{t}) = \frac{\kappa}{(2\pi\sigma^2)^{3/2} \cdot 2\pi(e^\kappa - e^{-\kappa})} \cdot e^{\kappa \hat{\mathbf{y}}_\mathrm{n}^\mathsf{T} \mathbf{R} \hat{\mathbf{x}}_\mathrm{n} - \frac{1}{2\sigma^2} \|\mathbf{y}_\mathrm{p} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t}\|_2^2} \tag{5.1}$$

where $\mathbf{y} = (\mathbf{y}_\mathrm{p}, \hat{\mathbf{y}}_\mathrm{n})$ is an oriented model-shape point that is assumed to be in correspondence with the oriented data-shape point, $\mathbf{x}$, and where σ^2 is the *variance* of the positional noise model and κ is the *concentration parameter* of the orientational noise model. The oriented model point, $\mathbf{y}$, is also a parameter of the joint distribution, with the model orientation, $\hat{\mathbf{y}}_\mathrm{n}$, being the *central direction* and the model position, $\mathbf{y}_\mathrm{p}$, being the *mean position*.

In the correspondence phase, the IMLOP algorithm selects the oriented point on the model shape that maximizes the match likelihood function of (5.1). This operation reduces to computing the oriented model point, $\mathbf{y} = (\mathbf{y}_\mathrm{p}, \hat{\mathbf{y}}_\mathrm{n})$, that minimizes

$$\mathrm{E}_\mathrm{IMLOP}(\mathbf{x}, \mathbf{y}, \sigma^2, \kappa, \mathbf{R}, \mathbf{t}) = \frac{1}{2\sigma^2} \|\mathbf{y}_\mathrm{p} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t}\|_2^2 - \kappa \hat{\mathbf{y}}_\mathrm{n}^\mathsf{T} \mathbf{R} \hat{\mathbf{x}}_\mathrm{n} \tag{5.2}$$

which is the *match error function* for the IMLOP algorithm.

In the registration phase, the IMLOP algorithm solves the transformation that maximizes the total match likelihood function as defined in (3.2) while using the

match likelihood function defined by (5.1). This operation simplifies to computing the transformation, $\mathbf{T}$, that minimizes the following *total match error function*

$$\mathbf{T} = \underset{[\mathbf{R},\mathbf{t}]}{\operatorname{argmin}} \left(\frac{1}{2\sigma^2} \sum_{i=1}^{n} \|\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t}\|_2^2 - \kappa \sum_{i=1}^{n} \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} \right) \tag{5.3}$$

which is the registration cost function for the IMLOP algorithm.

5.2 Algorithm Overview

In this section, a high-level overview of the IMLOP algorithm is described with the following sections describing the low-level implementations of each algorithmic phase. Algorithm 5.1 provides a summary of the IMLOP algorithm, which is referred to in the discussion that follows.

The noise parameters, κ and σ^2, are estimated from the residual match errors at each iteration. The variance of position error (σ^2) is simply estimated as the mean square distance between the matches divided by the spatial dimensionality. The concentration of orientation error (κ) is estimated using the following approximation

Algorithm 5.1. Iterative Most Likely Oriented Point (IMLOP)

input : Data shape: $\mathbf{X} = \{\mathbf{x}_i\} = \{(\mathbf{x}_{\mathrm{p}i}, \hat{\mathbf{x}}_{\mathrm{n}i})\}$
Model shape: Ψ
Initial noise-model parameters: κ_0 and σ_0^2
Initial transformation: $\mathbf{T}_0 = [\mathbf{R}_0, \mathbf{t}_0]$

output: Final transformation $\mathbf{T} = [\mathbf{R}, \mathbf{t}]$ that aligns $\mathbf{X}$ and Ψ

Initialize: $\mathbf{T} \leftarrow \mathbf{T}_0$, $\kappa \leftarrow \kappa_0$, $\sigma^2 \leftarrow \sigma_0^2$

while *not converged* **do**

Compute the most likely oriented-point correspondences:

$$\mathbf{y}_i = \underset{(\mathbf{y}_{\mathrm{p}}, \hat{\mathbf{y}}_{\mathrm{n}}) \in \Psi}{\operatorname{argmin}} \left(\frac{1}{2\sigma^2} \|\mathbf{y}_{\mathrm{p}} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} + \mathbf{t}\|_2^2 - \kappa \hat{\mathbf{y}}_{\mathrm{n}}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} \right), \quad i = 1..n$$

Register the oriented-point correspondences:

$$\mathbf{T} = \underset{[\mathbf{R}, \mathbf{t}]}{\operatorname{argmin}} \left(\frac{1}{2\sigma^2} \sum_{i=1}^{n} \|\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t}\|_2^2 - \kappa \sum_{i=1}^{n} \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} \right)$$

Update the noise-model parameters:

$$\kappa = \tfrac{\bar{R}(3-\bar{R}^2)}{1-\bar{R}^2}, \quad \sigma^2 = \tfrac{1}{3n} \textstyle\sum_{i=1}^{n} \|\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t})\|_2^2$$

end

to its maximum likelihood estimate[85,86]

$$\kappa \approx \frac{\bar{R}(3 - \bar{R}^2)}{1 - \bar{R}^2} \tag{5.4}$$

$$\bar{R} = \frac{1-w}{n} \sum_{i=1}^{n} \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} + \frac{w}{\alpha} \sum_{i=1}^{n} \mathbf{y}'_{\mathrm{p}i}{}^{\mathsf{T}} \mathbf{R} \mathbf{x}'_{\mathrm{p}i}\,, \tag{5.5}$$

$$\alpha = \sum_{i=1}^{n} \|\mathbf{y}'_{\mathrm{p}i}\| \cdot \|\mathbf{R}\mathbf{x}'_{\mathrm{p}i}\|\,,$$

$$\mathbf{x}'_{\mathrm{p}i} = \mathbf{x}_{\mathrm{p}i} - \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_{\mathrm{p}i}\,, \quad \mathbf{y}'_{\mathrm{p}\mathbf{i}} = \mathbf{y}_{\mathrm{p}i} - \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}_{\mathrm{p}i}$$

where w is a user-defined parameter that weights the relative importance of the orientation and position data terms used to estimate κ. The position data is included when estimating the distribution of the orientation match errors in order to prevent κ

from growing too large. For a closed shape model, it is often possible to find a nearly perfect orientation match for any given data orientation. Thus, estimating κ based on the match errors of the orientation elements alone may progressively over-estimate κ, which can cause a geometrically inconsistent set of match positions. To counter this, we consider the rotational misalignment represented in both the orientation and position elements of the data when computing $\bar{R}$ in (5.5), which balances κ at appropriate values. We equally weight the orientation- and position-element terms using $w = 0.5$. In light of the motivation above, it may be desirable to restrict the effect of position errors to only decrease orientation confidence. In this case, the following alternative estimate for $\bar{R}$ may be used

$$\bar{R} = \min\left(\frac{1}{n}\sum_{i=1}^{n}\hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}}\mathbf{R}\hat{\mathbf{x}}_{\mathrm{n}i}\,,\,\bar{R}'\right) \tag{5.6}$$

where $\bar{R}'$ represents the value of $\bar{R}$ computed by (5.5).

5.3 Correspondence Phase

This section describes an efficient implementation for the correspondence phase of the IMLOP algorithm. Namely, an approach is described to efficiently compute a most likely oriented match from the model shape, being the match that minimizes the match error function of (5.2). For the implementation described in this section,

it is assumed that the following alternative match error function is used

$$\mathrm{E}_{\mathrm{IMLOP}}(\mathbf{x}, \mathbf{y}, \sigma^2, \kappa, \mathbf{R}, \mathbf{t}) = \frac{1}{2\sigma^2}\|\mathbf{y}_{\mathrm{p}} - \mathbf{R}\mathbf{x}_{\mathrm{p}} - \mathbf{t}\|_2^2 + \kappa(1 - \hat{\mathbf{y}}_{\mathrm{n}}^{\mathsf{T}}\mathbf{R}\hat{\mathbf{x}}_{\mathrm{n}}) \tag{5.7}$$

which, unlike (5.2), is always positive by addition of an extra κ term.

IMLOP's search strategy for computing the most likely oriented-point correspondence from a model shape is based on a modification of the PD-tree search strategies presented earlier. Construction of IMLOP's PD tree follows the standard construction process described in Section 2.3 for positional feature data, but differs in that information regarding the orientational feature data are stored as additional parameters within each node. Thus, the addition of orientational features for the IMLOP algorithm does not modify the topology of the PD tree, but only affects the information stored by each node. This additional information includes the average datum orientation ($\hat{\mathbf{n}}_{\mathrm{avg}}$) and the maximum angular deviation ($\theta_{\max}$) from the average orientation within each node. Figure 5.1 illustrates this difference in the node parameters for the IMLOP algorithm compared to the standard PD tree described in Section 2.3.

$$\hat{\mathbf{n}}_{\mathrm{avg}} = \frac{\sum_{i \in Node} \hat{\mathbf{y}}_{\mathrm{n}i}}{\|\sum_{i \in Node} \hat{\mathbf{y}}_{\mathrm{n}i}\|} \tag{5.8}$$

$$\theta_{\max} = \max_{i \in Node} \left(\operatorname{acos}\left(\hat{\mathbf{n}}_{\mathrm{avg}}^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}i}\right)\right) \tag{5.9}$$

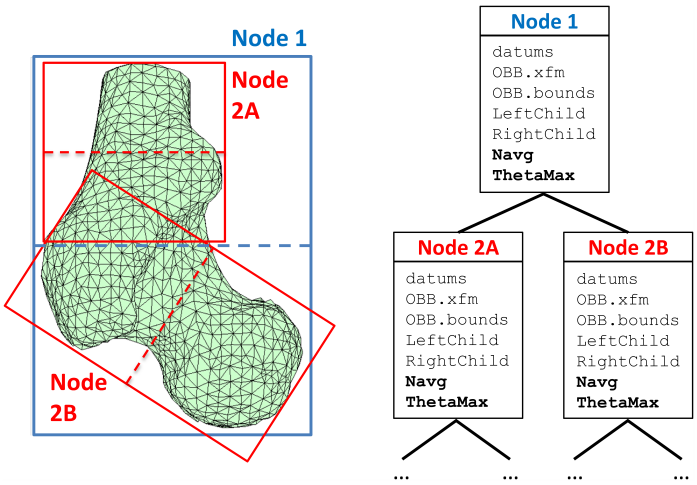

Figure 5.1: Illustration of nodes forming the upper two levels of a PD tree for the IMLOP algorithm, which incorporates datums having both positional and orientational feature elements. Compared to the standard PD tree described in Section 2.3 for datums having only positional feature elements, the topology of the PD tree remains the same, but the nodes store additional information regarding the datum orientations. These added node parameters are displayed in bold font in the figure.

Similar to previous discussions, a correspondence search for a given oriented data point, $\mathbf{x} = (\mathbf{x}_\mathrm{p}, \hat{\mathbf{x}}_\mathrm{n})$, begins with a current candidate for the best match, which has some match error, E_best. At each node in the tree, a fast test is required to determine whether any point within the node may produce a match error lower than E_best. Suppose a lower bound ($E_\mathrm{n,min}$) is known for the orientation component ($\kappa(1 - \hat{\mathbf{y}}_{\mathrm{n}i}^\mathsf{T}\mathbf{R}\hat{\mathbf{x}}_\mathrm{n})$) of the match error, being effective for all datum orientations, $\{\hat{\mathbf{y}}_{\mathrm{n}i}\}$, within a node. It follows from the match error function (5.7) that a better match within the node is only possible if $\mathbf{x}_\mathrm{p}$ lies within a distance of d_max from the node boundary, as defined by

$$d_\mathrm{max} = \sqrt{2\sigma^2(E_\mathrm{best} - E_\mathrm{n,min})} \ . \tag{5.10}$$

While a lower bound of $E_\mathrm{n,min} = 0$ could be used, a tighter bound is desirable to

maximize search efficiency.

Given the average datum orientation ($\hat{\mathbf{n}}_{\text{avg}}$) of a node and the maximum angular deviation ($\theta_{\max}$) from the average orientation, it follows that a lower bound on the angular difference between $\hat{\mathbf{x}}_{\text{n}}$ and any datum orientation, $\{\hat{\mathbf{y}}_{\text{n}i}\}$, that is contained within the node is given by

$$\theta_{\min} = \max\left(0\,,\, \operatorname{acos}\left(\hat{\mathbf{n}}_{\text{avg}}^{\mathsf{T}}\mathbf{R}\hat{\mathbf{x}}_{\text{n}}\right) - \theta_{\max}\right) \quad . \tag{5.11}$$

A tight lower bound on the orientation component of the match error for all orientations contained within the node is then obtained as

$$E_{\text{n,min}} = \kappa(1 - \cos(\theta_{\min})) \quad . \tag{5.12}$$

When performing the PD-tree search, nodes are skipped whenever $\mathbf{x}_{\text{p}}$ lies at a distance greater than $d_{\max}$ from the node's bounding box. Whenever a leaf node is searched, match errors are computed for every datum within the leaf node, and the candidate for the best match is updated when match errors lower than E_{best} are found. This PD-tree search strategy guarantees that the best match is always found and does not mis-match points near node boundaries, for example. This PD-tree search strategy is summarized as Algorithm 5.2.

Note that for a model shape represented by a mesh (rather than by a point cloud) the datum point, $\mathbf{y}_j = (\mathbf{y}_{\text{p}j}, \hat{\mathbf{y}}_{\text{n}j})$, that is used to compute the datum match error in

line 6 of Algorithm 5.2 is obtained by setting $\hat{\mathbf{y}}_{\mathrm{n}j}$ to the triangle normal and $\mathbf{y}_{\mathrm{p}j}$ to the point on the datum triangle that is closest by Euclidean distance to $\mathbf{x}_{\mathrm{p}}$.

Algorithm 5.2. IMLOP PD-Tree Node Search

input : Oriented data point being matched: $\mathbf{x} = (\mathbf{x}_{\mathrm{p}}, \hat{\mathbf{x}}_{\mathrm{n}})$
Data noise-model parameters: κ and σ^2
Current registration parameters: $[\mathbf{R}, \mathbf{t}]$
Current candidate for best match and its match error: $[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}]$
Node being searched: $\mathcal{N}$
output: Updated candidate for the best match: $[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}]$

```
1  Compute θ_min and d_max for this node based on N.n̂_avg and N.θ_max
2  B ← bounding box of N expanded by d_max in all directions
3  if B contains x_p then
4  |  if N is a leaf node then
5  |  |  foreach datum_j ∈ N do
6  |  |  |  Compute the match error for this datum:
   |  |  |      E_j ← E_IMLOP(x, y_j, σ^2, κ, R, t)        (Equ. (5.7))
7  |  |  |  if E_j < E_best then update the best match:
   |  |  |      [y_best, E_best] ← [y_j, E_j]
8  |  |  end
9  |  else
10 |  |  Search left and right child nodes of N
11 |  end
12 end
13 Return the updated best match: [y_best, E_best]
```

5.4 Registration Phase

This section presents a closed-form solution for the registration phase of the IMLOP algorithm, which is to solve the transformation, $\mathbf{T}$, that optimally aligns two corresponding oriented point sets characterized by isotropic noise, or, in other words, that minimizes the registration cost function of the IMLOP algorithm (5.3), which is

repeated below for ease-of-reference

$$\mathbf{T} = \underset{[\mathbf{R},\,\mathbf{t}]}{\operatorname{argmin}} \left(\frac{1}{2\sigma^2} \sum_{i=1}^{n} \|\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t}\|_2^2 - \kappa \sum_{i=1}^{n} \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} \right) \ .$$

This minimization turns out to be a special case of a more general form and solution proposed by Liu et al.[28] For sake of completeness, the solution for this special case is presented as Algorithm 5.3.

By recentering the point positions (as in (5.5)), the translation, $\mathbf{t}$, may be factored out of (5.3), enabling the rotation, $\mathbf{R}$, to be solved independently by minimizing

$$\begin{aligned} E &= \frac{1}{2\sigma^2} \sum_{i=1}^{n} \|\mathbf{y}'_{\mathrm{p}i} - \mathbf{R}\mathbf{x}'_{\mathrm{p}i}\|_2^2 - \kappa \sum_{i=1}^{n} \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} \\ &= \frac{1}{2\sigma^2} \left[\sum_{i=1}^{n} \|\mathbf{y}'_{\mathrm{p}i}\|_2^2 - 2 \sum_{i=1}^{n} {\mathbf{y}'_{\mathrm{p}i}}^{\mathsf{T}} \mathbf{R}\mathbf{x}'_{\mathrm{p}i} + \sum_{i=1}^{n} \|\mathbf{R}\mathbf{x}'_{\mathrm{p}i}\|_2^2 \right] - \kappa \sum_{i=1}^{n} \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} \end{aligned} \tag{5.13}$$

which is equivalent to maximizing

$$F = \frac{1}{\sigma^2} \sum_{i=1}^{n} {\mathbf{y}'_{\mathrm{p}i}}^{\mathsf{T}} \mathbf{R}\mathbf{x}'_{\mathrm{p}i} + \kappa \sum_{i=1}^{n} \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} \ . \tag{5.14}$$

The maximization of (5.14) is solved in closed-form using a modification of the SVD method of Arun.[32] The modification appears in line 2 of Algorithm 5.3, which involves modifying the cross-covariance matrix, $\mathbf{H}$, to incorporate a weighted sum of the orientation- and position-data elements, rather than defining the cross covariance based on the position data alone. Algorithm 5.3 provides a summary of the point-to-

point registration method for aligning the oriented point sets.

Algorithm 5.3. IMLOP Registration of Oriented-Point Matches

input : Oriented data-shape point set: $\mathbf{X} = \{(\mathbf{x}_{\mathrm{p}i}, \hat{\mathbf{x}}_{\mathrm{n}i})\}$
Oriented model-shape point set: $\mathbf{Y} = \{(\mathbf{y}_{\mathrm{p}i}, \hat{\mathbf{y}}_{\mathrm{n}i})\}$
Data-shape noise parameters: κ and σ^2

output: Transformation $\mathbf{T} = [\mathbf{R}, \mathbf{t}]$ that aligns $\mathbf{X}$ and $\mathbf{Y}$

1 Recenter the oriented-point positions:
$\mathbf{x}'_{\mathrm{p}i} \leftarrow \mathbf{x}_{\mathrm{p}i} - \bar{\mathbf{x}}_{\mathrm{p}}$, $\mathbf{y}'_{\mathrm{p}i} \leftarrow \mathbf{y}_{\mathrm{p}i} - \bar{\mathbf{y}}_{\mathrm{p}}$
where $\bar{\mathbf{x}}_{\mathrm{p}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_{\mathrm{p}i}$, $\bar{\mathbf{y}}_{\mathrm{p}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_{\mathrm{p}i}$

2 Compute the cross-covariance matrix: $\mathbf{H} \leftarrow \kappa\mathbf{H_1} + \frac{1}{\sigma^2}\mathbf{H_2}$
where $\mathbf{H_1} = \sum_{i=1}^{n} \mathbf{x}'_{\mathrm{p}i}\mathbf{y}'^{\mathsf{T}}_{\mathrm{p}i}$, $\mathbf{H_2} = \sum_{i=1}^{n} \hat{\mathbf{x}}_{\mathrm{n}i}\hat{\mathbf{y}}^{\mathsf{T}}_{\mathrm{n}i}$

3 Compute rotation from SVD of $\mathbf{H}$:
$\mathbf{R} \leftarrow \mathbf{V}\mathbf{U}^{\mathsf{T}}$ where $\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathsf{T}}$

4 **if** $\det(\mathbf{R}) = -1$ **then** $\mathbf{R} \leftarrow \mathbf{V}'\mathbf{U}^{\mathsf{T}}$
where $\mathbf{V} = [\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}]$, $\mathbf{V}' = [\mathbf{v_1}, \mathbf{v_2}, -\mathbf{v_3}]$

5 Compute translation: $\mathbf{t} \leftarrow \bar{\mathbf{y}}_{\mathrm{p}} - \mathbf{R}\bar{\mathbf{x}}_{\mathrm{p}}$

5.5 Experimental Results and Discussion

In this section, the IMLOP algorithm is evaluated by registration experiments involving a mesh model of a human femur (Figure 5.2B) created by segmentation from CT imaging. The registration performance of IMLOP is compared to that of the ICP algorithm. Further experimental evaluation of the IMLOP algorithm also appears in the following chapter (Section 6.5).

In all studies, a common set of termination criteria is used for each evaluated algorithm, being when the magnitude of change in the estimated transformation, $\mathbf{T} = [\mathbf{R}, \mathbf{t}]$, falls below 0.001 mm and 0.001 degrees for two consecutive iterations or

when 200 iterations is reached.

5.5.1 Experiment 1: Registration with Isotropic Noise

In this experiment, the performance of the proposed IMLOP algorithm is evaluated for registering data characterized by isotropic noise; comparison is made with the ICP algorithm.

Registrations were conducted by generating data shapes from the darkly shaded subregion of the femur-bone mesh (Figure 5.2B) using uniform random sampling and applying isotropic Gaussian noise to the samples. The data shapes were misaligned from the mesh model by random spatial transformations generated on the interval $[10, 20]$ mm and $[10, 20]$ degrees and then registered back to the model using the same data and misalignment for each algorithm. In this manner, a precise comparison of registration accuracy can be assessed, since the ground-truth registration is exactly known.

Registration accuracy was evaluated by treating the center points of each triangle in the sampled region of the mesh (i.e., in the darkly shaded region) as validation points. The average distance between the registered and ground-truth positions of these validation points defines the positional target registration error (TRE). An orientational TRE is similarly defined as the average angle between the registered

and ground-truth surface-normal orientations of the validation points.

Each algorithm autonomously decides whether to *accept* or *reject* each registration outcome by using the final match residuals to compute a measure of "confidence" in the registration accuracy. For ICP, a registration is rejected if the average match distance exceeds a user-controlled threshold. For IMLOP, the reject criterion is extended by an "or" condition to include a second user-controlled maximum threshold on the average orientation match error.

Registrations were performed with data-shape sample sizes of 10, 20, 35, 50, 75, and 100 points. Results are presented for noise levels of 1 mm (1 degree) and 2 mm (2 degrees) standard deviation of Gaussian (wrapped-Gaussian) noise applied to the data-point positions (orientations). For each sample-size / noise-level pair, 300 randomized trials were conducted, generating a different data shape and misalignment for each trial.

The average TREs of the accepted registration trials and the registration rejection rates of ICP and IMLOP are plotted in Figures 5.3 and 5.4, respectively, for each sample size and noise level. For these plots, the match-error thresholds for accepting versus rejecting a registration outcome were set to twice the standard deviation of applied noise. Plots generated using different threshold values show similar results.

A "zoomed-in" view of the registration outcomes for one sample size (75 points) and noise level (1 mm and 1 degree standard deviation) is shown in Figure 5.2, which plots the TRE and final match residuals for each of the 300 randomized trials. In

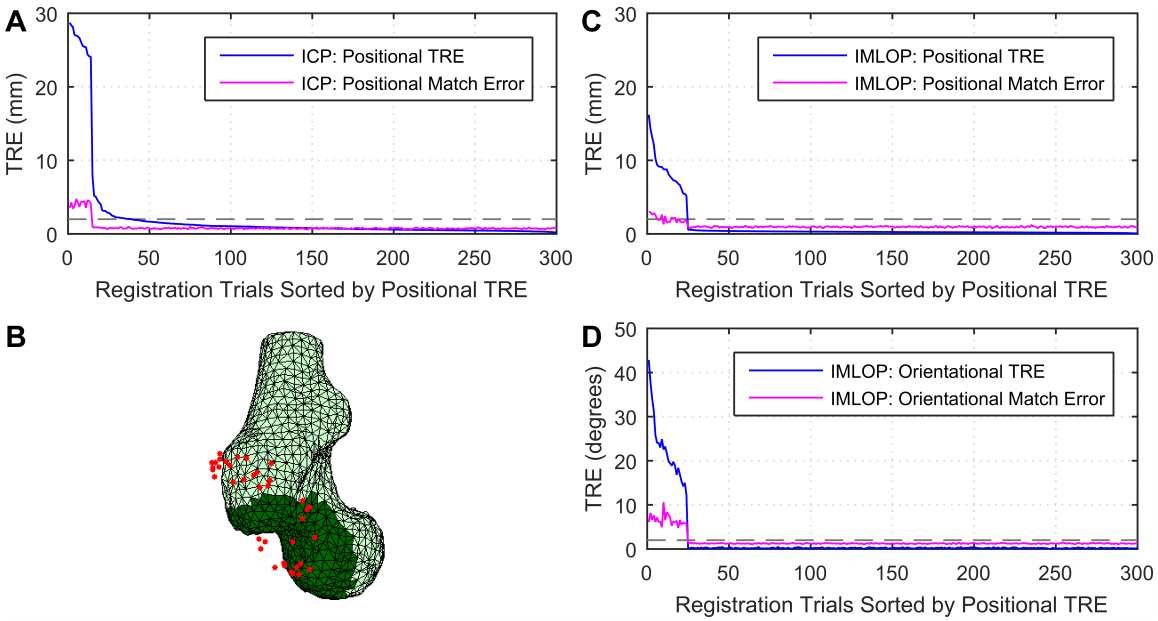

Figure 5.2: Experiment 1: (B) human femur model used in the registration experiments showing a randomly generated and misaligned data shape. TREs and final match errors are plotted for each of the 300 randomized trials from the isotropic noise study involving 75 samples and 1 mm (1 degree) standard deviation of noise; trials are sorted on the x-axis by positional TRE for (A) ICP and (C,D) IMLOP.

order to better appreciate the relationship between the final match residuals and the actual TRE, the trials have been sorted along the x-axis by positional TRE for each algorithm.

As seen in Figure 5.3, the IMLOP algorithm achieves substantially improved registration accuracy compared to ICP for all sample sizes and noise levels studied. The plots of Figure 5.4 further indicate that IMLOP robustly rejects the inaccurate registration outcomes, which increase in frequency at small sample sizes, whereas ICP rarely detects the inaccurate outcomes. This observation is further highlighted by inspection of Figure 5.2. In this figure, the rejection thresholds are plotted as horizontal dashed lines, with the rejected trials being those having a match error above

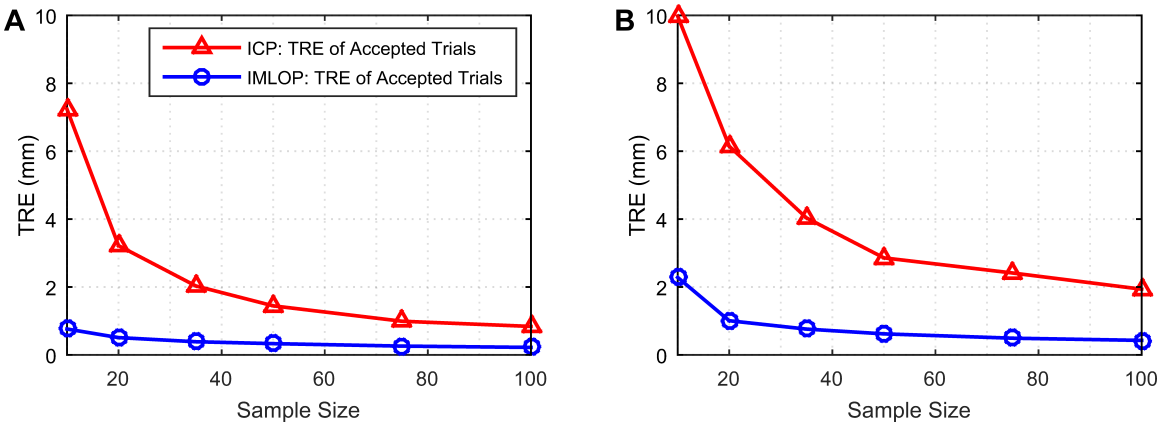

Figure 5.3: Experiment 1: average TREs of the "accepted" registration outcomes for the IMLOP and ICP algorithms are plotted by sample size for noise levels of (A) 1 mm (1 degree) and (B) 2 mm (2 degrees) standard deviation.

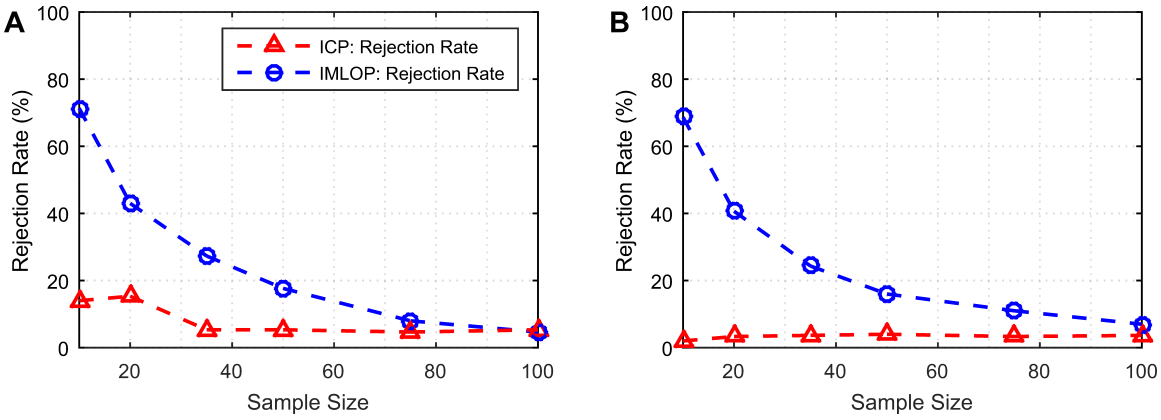

Figure 5.4: Experiment 1: registration rejection rates for the IMLOP and ICP algorithms are plotted by sample size for noise levels of (A) 1 mm (1 degree) and (B) 2 mm (2 degrees) standard deviation.

any threshold line. It can be seen that the TREs for IMLOP decrease sharply for registrations terminating near the correct alignment (thus, high registration accuracy), which is closely paralleled by a sharp decrease in the position and orientation match errors (thus, robust rejection of inaccurate outcomes). The TREs for ICP, on the other hand, decrease gradually within the vicinity of the correct alignment (thus, low registration accuracy), and ICP fails to reject many inaccurate outcomes (thus, non-robust rejection of inaccurate outcomes).

5.6 Concluding Remarks

The IMLOP registration algorithm incorporates position and orientation information of the shapes being registered within a cohesive, probabilistic framework, leading to substantial improvement in registration accuracy over the position-based ICP algorithm when registering oriented data. In addition, the proposed IMLOP algorithm offers a robust mechanism to autonomously characterize a registration outcome as likely to be accurate vs. inaccurate, unlike ICP. Importantly, an efficient implementation for IMLOP has been developed, including a fast PD-tree-based search strategy for computing the most likely matches from the model shape that accounts for the orientation component of the match likelihood. Further evaluation of the IMLOP algorithm is found in the experiments of the following chapter.

5.7 Contributions

The contributions from this chapter include:

3. Iterative Most Likely Oriented Point (IMLOP) Algorithm[26,27]

   (a) a probabilistic algorithm for registering *positional* and *orientational* feature data (i.e., oriented points) that are characterized by *isotropic* uncertainty; the algorithm incorporates a dynamically adaptive noise model to aptly weight the relative confidence in the position versus orientation data within each iteration

   (b) an efficient implementation of the IMLOP algorithm consisting of:

      i. *IMLOP Correspondence Phase*: a fast PD-tree-based search for efficiently computing the most likely matches from a model shape considering both the position and orientation components of the match error

      ii. *IMLOP Registration Phase*: a closed-form solution to the problem of registering corresponding oriented point sets assuming isotropic uncertainty in the positions and orientations; this solution is a special case of a more general solution found in [28]

   (c) a mechanism for autonomously assessing a registration outcome in order to determine, with high confidence, whether a registration has succeeded or failed to compute an accurate shape alignment

5.8 Published Work

Material from this chapter has appeared in the following publications:

1. S. D. Billings and R. H. Taylor, "Generalized iterative most likely oriented-point (G-IMLOP) registration," *International Journal of Computer Assisted Radiology and Surgery*, vol. 10, no. 8, pp. 1213–1226, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11548-015-1221-2

2. S. Billings and R. Taylor, "Iterative most likely oriented point registration," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2014*, ser. Lecture Notes in Computer Science, P. Golland, N. Hata, C. Barillot, J. Hornegger, and R. Howe, Eds. Springer International Publishing, 2014, vol. 8673, pp. 178–185. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10404-1_23

Chapter 6

Generalized Iterative Most Likely Oriented Point (G-IMLOP) Algorithm

This chapter describes the Generalized Iterative Most Likely Oriented Point (G-IMLOP) algorithm, which extends the IMLOP algorithm described in Chapter 5 by incorporating anisotropic noise models for both the position and orientation elements. Thus, G-IMLOP is an algorithm for registering oriented feature data characterized by anisotropic uncertainty in the position and/or orientation data. Similar to the IMLOP algorithm, G-IMLOP is an ICP-based method that incorporates a probabilistic framework to combine the position and orientation information of the shapes being registered. Content of this chapter has appeared in [27].

The anisotropic extension enables G-IMLOP to achieve higher registration accuracy when registering data characterized by anisotropic noise in the measured positions and/or orientations. The experiments in this chapter assess both G-IMLOP and the previously described IMLOP algorithm under anisotropic noise conditions. Sources of anisotropic measurement uncertainties in clinical data are discussed in the introduction to Chapter 4. A discussion regarding the use of orientation data in clinical applications and of relevant prior works is found in the introduction to Chapter 5.

6.1 Probabilistic Model

The G-IMLOP algorithm incorporates a probabilistic framework formulated using anisotropic Gaussian and Kent distributions to model the measurement errors of the position and orientation data, respectively. In like manner to the isotropic Fisher distribution, the anisotropic Kent distribution is the analog on the unit sphere of a multivariate Gaussian distribution with unconstrained (anisotropic) covariance[85] (see Appendix D for a description and example illustrations of the Kent distribution). Assuming zero-mean, independent, identically distributed (iid) error for both the orientation- and position-based data, the *match likelihood function* for an oriented data point, $\mathbf{x} = (\mathbf{x}_{\mathrm{p}}, \hat{\mathbf{x}}_{\mathrm{n}})$, that has been transformed by a current registration estimate,

$[\mathbf{R},\, \mathbf{t}]$, is defined as

$$f_{\text{match}}(\mathbf{x}\,;\, \mathbf{y}, \boldsymbol{\Sigma}, \kappa, \beta, \hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\gamma}}_2, \mathbf{R}, \mathbf{t}) = \frac{1}{(2\pi)^{3/2}|\boldsymbol{\Sigma}|^{1/2} \cdot \mathrm{c}(\kappa, \beta)} \cdot e^{\kappa \hat{\mathbf{y}}_{\mathrm{n}}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}} + \beta\left((\hat{\boldsymbol{\gamma}}_1^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}})^2 - (\hat{\boldsymbol{\gamma}}_2^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}})^2\right) - \frac{1}{2}(\mathbf{y}_{\mathrm{p}} - \mathbf{R}\mathbf{x}_{\mathrm{p}} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{\mathrm{p}} - \mathbf{R}\mathbf{x}_{\mathrm{p}} - \mathbf{t})} \tag{6.1}$$

where $\mathbf{y} = (\mathbf{y}_{\mathrm{p}}, \hat{\mathbf{y}}_{\mathrm{n}})$ is an oriented model-shape point that is assumed to be in correspondence with the oriented data-shape point, $\mathbf{x}$. $\boldsymbol{\Sigma}$ is the *covariance matrix* of the positional noise model and the orientation noise model is defined by the *concentration parameter*, κ, the *ellipticity parameter*, β $(0 \leq 2\beta < \kappa)$, and the *major* and *minor axes*, $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$, which define the directions of the elliptical level sets of the Kent distribution on the unit sphere. The major and minor axes are orthogonal to each other and to the *central direction*, which is $\hat{\mathbf{y}}_{\mathrm{n}}$. Similarly, $\mathbf{y}_{\mathrm{p}}$ is the *mean position*. The ellipticity parameter, β, controls the amount of anisotropy, which increases with larger values. When β equals zero, the expression reduces to the isotropic Fisher distribution used by the the IMLOP algorithm (Chapter 5). Finally, $\mathrm{c}(\kappa, \beta)$ is the normalizing constant of the Kent distribution, involving a complex expression of modified Bessel functions.[85]

In the correspondence phase, the G-IMLOP algorithm selects the oriented point on the model shape that maximizes the match likelihood function of (6.1). This op-

eration reduces to computing the oriented model point, $\mathbf{y} = (\mathbf{y}_\mathrm{p}, \hat{\mathbf{y}}_\mathrm{n})$, that minimizes

$$\begin{aligned} \mathrm{E}_{\text{G-IMLOP}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\Sigma}, \kappa, \beta, \hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\gamma}}_2, \mathbf{R}, \mathbf{t}) = \frac{1}{2}(\mathbf{y}_\mathrm{p} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t})^\mathsf{T}\mathbf{R}\boldsymbol{\Sigma}^{-1}\mathbf{R}^\mathsf{T}(\mathbf{y}_\mathrm{p} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t}) \\ - \kappa\hat{\mathbf{y}}_\mathrm{n}^\mathsf{T}\mathbf{R}\hat{\mathbf{x}}_\mathrm{n} - \beta\left((\hat{\boldsymbol{\gamma}}_1^\mathsf{T}\mathbf{R}\hat{\mathbf{x}}_\mathrm{n})^2 - (\hat{\boldsymbol{\gamma}}_2^\mathsf{T}\mathbf{R}\hat{\mathbf{x}}_\mathrm{n})^2\right) \quad . \quad (6.2) \end{aligned}$$

Since the major and minor axes, $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$, are perpendicular to the central direction, $\hat{\mathbf{y}}_\mathrm{n}$, this formulation would require recomputing $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$ for every oriented model point $(\mathbf{y}_\mathrm{p}, \hat{\mathbf{y}}_\mathrm{n})$ that is tested. Thus, in order to maximize the efficiency of the correspondence phase, in practice the following equivalent formulation is used as the *match error function* for the G-IMLOP algorithm

$$\begin{aligned} \mathrm{E}_{\text{G-IMLOP}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\Sigma}, \kappa, \beta, \hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\gamma}}_2, \mathbf{R}, \mathbf{t}) = \frac{1}{2}(\mathbf{y}_\mathrm{p} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t})^\mathsf{T}\mathbf{R}\boldsymbol{\Sigma}^{-1}\mathbf{R}^\mathsf{T}(\mathbf{y}_\mathrm{p} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t}) \\ - \kappa\hat{\mathbf{y}}_\mathrm{n}^\mathsf{T}\mathbf{R}\hat{\mathbf{x}}_\mathrm{n} - \beta\left((\hat{\boldsymbol{\gamma}}_1^\mathsf{T}\mathbf{R}^\mathsf{T}\hat{\mathbf{y}}_\mathrm{n})^2 - (\hat{\boldsymbol{\gamma}}_2^\mathsf{T}\mathbf{R}^\mathsf{T}\hat{\mathbf{y}}_\mathrm{n})^2\right) \quad (6.3) \end{aligned}$$

where $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$ are redefined to be perpendicular to the data orientation, $\hat{\mathbf{x}}_\mathrm{n}$, rather than perpendicular to the central direction, $\hat{\mathbf{y}}_\mathrm{n}$. With this formulation, $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$ are thus defined only once with respect to the measured data and need not be recomputed for every oriented model point that is tested. This reduces the computational overhead of the correspondence phase.

In the registration phase, the G-IMLOP algorithm solves the transformation that maximizes the total match likelihood function (3.2). This operation simplifies to

computing the transformation, $\mathbf{T}$, that minimizes the following *total match error function*

$$\mathbf{T} = \underset{[\mathbf{R},\,\mathbf{t}]}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n} (\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_i^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t}) \right. \\ \left. - \sum_{i=1}^{n} \left(\kappa_i \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} + \beta_i \left[(\hat{\boldsymbol{\gamma}}_{1i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{n}i})^2 - (\hat{\boldsymbol{\gamma}}_{2i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{n}i})^2 \right] \right) \right) \quad (6.4)$$

where the redefinition of the major and minor axes, $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$, as described above for the correspondence phase has been retained here as well for the registration phase. This is the registration cost function for the G-IMLOP algorithm.

6.2 Algorithm Overview

In this section, a high-level overview of the G-IMLOP algorithm is described, followed by sections detailing the low-level implementations of each algorithmic phase. Algorithm 6.1 provides a summary of the G-IMLOP algorithm. Note that, for G-IMLOP, the noise-model parameters are specified as inputs to the algorithm, being defined independently for each data point.

Algorithm 6.1. Generalized Iterative Most Likely Oriented Point (G-IMLOP)

input : Data shape: $\mathbf{X} = \{\mathbf{x}_i\} = \{(\mathbf{x}_{\mathrm{p}i}, \hat{\mathbf{x}}_{\mathrm{n}i})\}$
Model shape: Ψ
Data-shape noise-model parameters: $\{(\boldsymbol{\Sigma}_i, \kappa_i, \beta_i, \hat{\boldsymbol{\gamma}}_{1i}, \hat{\boldsymbol{\gamma}}_{2i})\}$
Initial transformation: $\mathbf{T}_0 = [\mathbf{R}_0, \mathbf{t}_0]$

output: Final transformation $\mathbf{T}$ that aligns $\mathbf{X}$ and Ψ

Initialize: $\mathbf{T} \leftarrow \mathbf{T}_0$

while *not converged* **do**

Compute the oriented-point correspondences:

$$\mathbf{y}_i = \underset{(\mathbf{y}_{\mathrm{p}}, \hat{\mathbf{y}}_{\mathrm{n}}) \in \Psi}{\operatorname{argmin}} \left(\frac{1}{2} (\mathbf{y}_{\mathrm{p}} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} + \mathbf{t})^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_i^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{\mathrm{p}} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} + \mathbf{t}) \right.$$
$$\left. - \kappa_i \hat{\mathbf{y}}_{\mathrm{n}}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} - \beta_i \left((\hat{\boldsymbol{\gamma}}_{1i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{n}})^2 - (\hat{\boldsymbol{\gamma}}_{2i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{n}})^2 \right) \right), \quad i = 1..n$$

Register the oriented-point correspondences:

$$\mathbf{T} = \underset{[\mathbf{R}, \mathbf{t}]}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n} (\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_i^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t}) \right.$$
$$\left. - \sum_{i=1}^{n} \left(\kappa_i \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} + \beta_i \left[(\hat{\boldsymbol{\gamma}}_{1i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{n}i})^2 - (\hat{\boldsymbol{\gamma}}_{2i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{n}i})^2 \right] \right) \right)$$

end

6.3 Correspondence Phase

This section describes an efficient implementation for the correspondence phase of the G-IMLOP algorithm. Namely, an approach is described to efficiently compute a most likely oriented match from the model shape, being the match that minimizes the match error function of (6.3). For the implementation described in this section, the following alternative match error function is used

$$\begin{aligned} \mathrm{E}_{\text{G-IMLOP}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\Sigma}, \kappa, \beta, \hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\gamma}}_2, \mathbf{R}, \mathbf{t}) = \frac{1}{2}(\mathbf{y}_\mathrm{p} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t})^\mathsf{T}\mathbf{R}\boldsymbol{\Sigma}_i^{-1}\mathbf{R}^\mathsf{T}(\mathbf{y}_\mathrm{p} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t}) \\ + \kappa(1 - \hat{\mathbf{y}}_\mathrm{n}^\mathsf{T}\mathbf{R}\hat{\mathbf{x}}_\mathrm{n}) - \beta\left((\hat{\boldsymbol{\gamma}}_1^\mathsf{T}\mathbf{R}^\mathsf{T}\hat{\mathbf{y}}_\mathrm{n})^2 - (\hat{\boldsymbol{\gamma}}_2^\mathsf{T}\mathbf{R}^\mathsf{T}\hat{\mathbf{y}}_\mathrm{n})^2\right) \quad (6.5) \end{aligned}$$

which, unlike (6.3), is always positive by addition of an extra κ term (see Appendix E for justification).

A PD-tree search strategy is again employed in order to compute the oriented model point, $\mathbf{y}$, that minimizes G-IMLOP's match error function (6.5) for a given oriented data point, $\mathbf{x}$. For G-IMLOP, the problem is complicated by the anisotropy in both the position and orientation components of the match error.

G-IMLOP's PD-tree search strategy incorporates a simplification of the ideas developed in Chapter 4 for the anisotropic position-based matching of the IMLP algorithm and extends the ideas developed in Chapter 5 for incorporating orientation-based matching for the IMLOP algorithm, with the extension being to consider anisotropic uncertainty in the orientation data, whereas Chapter 5 assumed isotropic

uncertainty.

The PD tree for G-IMLOP is constructed in the same manner as the PD tree for the IMLOP algorithm, which was previously described in Section 5.3. The derivation of G-IMLOP's PD-tree search strategy also proceeds in similar manner to the IMLOP algorithm. First, a lower bound on the orientation component of the match error is established, which is used to form an upper-bound on the positional component of the match error, which is used for ruling out nodes of the PD tree. To illustrate this search procedure, suppose that a lower bound ($E_{\text{n,min}}$) is known for the orientation component of the match error, being effective for all datum orientations ($\{\hat{\mathbf{y}}_{\text{n}i}\}$) within a node; it follows from the match error equation (6.5) that a better match candidate within the node is only possible if the inequality

$$(\mathbf{y}_{\text{p}} - \mathbf{R}\mathbf{x}_{\text{p}} - \mathbf{t})^{\mathsf{T}}\mathbf{R}\boldsymbol{\Sigma}_i^{-1}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{\text{p}} - \mathbf{R}\mathbf{x}_{\text{p}} - \mathbf{t}) \leq 2(E_{\text{best}} - E_{\text{n,min}}) \tag{6.6}$$

is satisfied for some model-point position, $\mathbf{y}_{\text{p}}$, located within the bounds of the node, where E_{best} denotes the match error of the current candidate for the best match. The inequality of (6.6) defines the equation of an ellipsoid (and its inner area) centered at the transformed data-point position, $(\mathbf{R}\mathbf{x}_{\text{p}} + \mathbf{t})$. The node test for the PD-tree search is therefore to determine whether the ellipsoid defined by (6.6) intersects the bounding box of the node. If no intersection exists, then the node is ruled out of the search.

As for the IMLOP algorithm, the strategy for computing the lower bound on the orientation component of match error ($E_{\text{n,min}}$) for a node uses the average datum orientation, $\hat{\mathbf{n}}_{\text{avg}}$, of the node and the maximum angular deviation, $\theta_{\max}$, from the average orientation. Due to the anisotropic orientation uncertainty, however, the techniques applied for the IMLOP algorithm do not fully apply here and require modification for the G-IMLOP algorithm.

For the sake of illustration, consider a node having $\hat{\mathbf{n}}_{\text{avg}}$ located at the pole of the unit sphere depicted in Figure 6.1. Also consider that all orientations that are offset from $\hat{\mathbf{n}}_{\text{avg}}$ by $\theta_{\max}$ form a latitudinal ring on the unit sphere, within which all datum orientations of the node must be located. It is clear that if a transformed data-point orientation, $\mathbf{R}\hat{\mathbf{x}}_{\text{n}}$, is displaced by more than $\theta_{\max}$ from $\hat{\mathbf{n}}_{\text{avg}}$, then there must exist some orientation located on the latitudinal bounding ring that establishes a lower bound on the orientation match error for the node. The task then is to determine the orientation ($\mathbf{y}_{\text{n,ring}}$) on the bounding ring that establishes this lower bound. Due to the elliptical level sets of the Kent distribution centered at $\mathbf{R}\hat{\mathbf{x}}_{\text{n}}$, the naive solution of choosing the point on the latitudinal ring that intersects the arc connecting $\mathbf{R}\hat{\mathbf{x}}_{\text{n}}$ to $\hat{\mathbf{n}}_{\text{avg}}$ is non-optimal. However, the naive solution is generally close to optimal. Therefore, the approach is to initialize $\mathbf{y}_{\text{n,ring}}$ to the naive solution and then perform an efficient 1D Brent search[31] around the ring boundary in order to locate the point of minimal orientation match error that establishes a lower bound ($E_{\text{n,min}}$) on the orientation match error of the node.

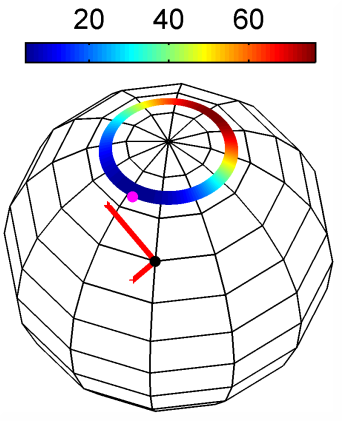

Figure 6.1: Unit sphere illustrating the problem of computing a lower bound on the anisotropic orientation match error of a Kent distribution for a node of the PD tree. The pole represents the average node orientation, $\hat{\mathbf{n}}_{\text{avg}}$. The latitudinal ring denotes all orientations offset by $\theta_{\max} = 15^\circ$ from $\hat{\mathbf{n}}_{\text{avg}}$, within which all datum orientations, $\hat{\mathbf{y}}_{\text{n}i}$, of the node are bounded. The data orientation, $\hat{\mathbf{x}}_{\text{n}}$, is indicated by a black dot with the major ($\hat{\gamma}_1$) and minor ($\hat{\gamma}_2$) axes of its Kent noise model shown as long and short red lines, respectively. The pink dot denotes the orientation, $\mathbf{y}_{\text{n,ring}}$, located on the latitudinal ring that provides a lower bound on the orientation match error for the node. Note that $\mathbf{y}_{\text{n,ring}}$ is not located on the arc that directly connects $\hat{\mathbf{x}}_{\text{n}}$ to $\hat{\mathbf{n}}_{\text{avg}}$ (i.e., the naive solution).

In summary, the full procedure for ruling out a node of the PD tree is to first compute a lower bound on the orientation match error ($E_{\text{n,min}}$) of the node using a 1D Brent search and then compute the intersection between the ellipsoid defined by (6.6) and the oriented bounding box of the node. As noted in a prior chapter, an efficient method for performing the ellipsoid-node intersection test is described by Larsson.[67] Algorithm 6.2 summarizes this PD-tree search procedure for the G-IMLOP algorithm.

Note that for a model shape represented by a mesh (rather than by a point cloud) the datum point, $\mathbf{y}_j = (\mathbf{y}_{\text{p}j}, \hat{\mathbf{y}}_{\text{n}j})$, that is used to compute the datum match error in line 6 of Algorithm 6.2 is obtained by setting $\hat{\mathbf{y}}_{\text{n}j}$ to the triangle normal and $\mathbf{y}_{\text{p}j}$ to

the point on the datum triangle that is the most likely match by position from $\mathbf{x}_\mathrm{p}$, which is discussed in Appendix B.

Algorithm 6.2. G-IMLOP PD-Tree Node Search

input : Oriented data point being matched: $\mathbf{x} = (\mathbf{x}_\mathrm{p}, \hat{\mathbf{x}}_\mathrm{n})$
Data noise-model parameters: $\mathbf{\Sigma}$, κ, β, $\hat{\boldsymbol{\gamma}}_1$, and $\hat{\boldsymbol{\gamma}}_2$
Current registration parameters: $[\mathbf{R}, \mathbf{t}]$
Current candidate for best match and its match error: $[\mathbf{y}_\mathrm{best}, E_\mathrm{best}]$
Node being searched: $\mathcal{N}$

output: Updated candidate for the best match: $[\mathbf{y}_\mathrm{best}, E_\mathrm{best}]$

Compute a lower bound, $E_\mathrm{n,min}$, on the orientation match error for the node using a 1D Brent search

Compute an intersection test between the ellipsoid

$$\mathcal{E} = \{\mathbf{z} \mid (\mathbf{z} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t})^\mathsf{T}\mathbf{R}\mathbf{\Sigma}^{-1}\mathbf{R}^\mathsf{T}(\mathbf{z} - \mathbf{R}\mathbf{x}_\mathrm{p} - \mathbf{t}) \leq 2(E_\mathrm{best} - E_\mathrm{n,min})\}$$

and the oriented bounding box of the node, $\mathcal{N}.OBB$

if $\mathcal{E}$ *intersects* $\mathcal{N}.OBB$ **then**
  if $\mathcal{N}$ *is a leaf node* **then**
    foreach $datum_j \in \mathcal{N}$ **do**
      Compute the match error for this datum:
        $E_j \leftarrow \mathrm{E}_\text{G-IMLOP}(\mathbf{x}, \mathbf{y}_j, \mathbf{\Sigma}, \kappa, \beta, \hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\gamma}}_2)$ (Equ. (6.5), Appendix B)
      if $E_j < E_{best}$ **then** update the best match:
        $[\mathbf{y}_\mathrm{best}, E_\mathrm{best}] \leftarrow [\mathbf{y}_j, E_j]$
    end
  else
    Search left and right child nodes of $\mathcal{N}$
  end
end
Return the updated best match: $[\mathbf{y}_\mathrm{best}, E_\mathrm{best}]$

6.4 Registration Phase

This section presents an implementation for the registration phase of the G-IMLOP algorithm, which solves the transformation, $\mathbf{T}$, that optimally aligns two cor-

responding oriented point sets characterized by anisotropic noise, or, in other words, that minimizes the registration cost function of the G-IMLOP algorithm (6.4), which is repeated below for ease-of-reference

$$\mathbf{T} = \underset{[\mathbf{R},\,\mathbf{t}]}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n} (\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_i^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{\mathrm{p}i} - \mathbf{R}\mathbf{x}_{\mathrm{p}i} - \mathbf{t}) \right.$$
$$\left. - \sum_{i=1}^{n} \left(\kappa_i \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}} \mathbf{R} \hat{\mathbf{x}}_{\mathrm{n}i} + \beta_i \left[(\hat{\boldsymbol{\gamma}}_{1i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{n}i})^2 - (\hat{\boldsymbol{\gamma}}_{2i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{n}i})^2 \right] \right) \right)$$

Unlike the closed-form solution available for IMLOP, this nonlinear cost function requires an iterative optimization approach. For the implementation described here, the BFGS quasi-Newton solver of the dlib open-source C++ software library[87] was used.

In order to apply the unconstrained quasi-Newton solver to minimize (6.4), it is necessary to first re-parameterize the variables being optimized in order to enforce the algebraic constraints of the rotation matrix, namely $\mathbf{R}^{\mathsf{T}}\mathbf{R} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$. To accomplish this, the Rodrigues parameterization is used, which represents rotation as a 3-vector, $\boldsymbol{a} = [a_x, a_y, a_z]$, whose direction and magnitude signify the axis and angular extent of rotation, respectively. The notation $\mathrm{R}(\boldsymbol{a})$ is used to signify the

3×3 rotation matrix corresponding to the Rodrigues vector, $\boldsymbol{a}$, which is defined as

$$\mathrm{R}(\boldsymbol{a}) = \mathbf{I} + \sin(\theta)\,\mathrm{skew}(\boldsymbol{\alpha}) + (1 - \cos(\theta))\,\mathrm{skew}(\boldsymbol{\alpha})^2 \tag{6.7}$$

$$\theta = \|\boldsymbol{a}\|_2 \tag{6.8}$$

$$\boldsymbol{\alpha} = \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|} \tag{6.9}$$

where θ is the magnitude of $\boldsymbol{a}$, representing the angle of rotation, and $\boldsymbol{\alpha}$ is the unit vector in the direction of $\boldsymbol{a}$, representing the axis of rotation. $\mathrm{skew}(\boldsymbol{\alpha})$ is the skew-symmetric matrix formed from the elements of $\boldsymbol{\alpha}$.

The transformation computed by each registration phase is solved as an incremental update, $\mathbf{\Delta T}$, on the prior transformation. Thus, the objective function that is minimized is the following

$$\mathbf{\Delta T} = \underset{[\mathrm{R}(\mathbf{\Delta}\boldsymbol{a}),\mathbf{\Delta t}]}{\operatorname{argmin}} \sum_{i=1}^{n} (C_{\mathrm{p}i} + C_{\mathrm{n}i}) \tag{6.10}$$

$$C_{\mathrm{p}i} = \frac{1}{2}\mathbf{z}_i^{\mathsf{T}}\mathbf{R}\mathbf{\Sigma}_i^{-1}\mathbf{R}^{\mathsf{T}}\mathbf{z}_i \tag{6.11}$$

$$\text{where} \quad \mathbf{z}_i = \mathrm{R}(\mathbf{\Delta}\boldsymbol{a})^{\mathsf{T}}(\mathbf{y}_{\mathrm{p}i} - \mathrm{R}(\mathbf{\Delta}\boldsymbol{a})(\mathbf{R}\mathbf{x}_{\mathrm{p}i} + \mathbf{t}) - \mathbf{\Delta t})$$

$$C_{\mathrm{n}i} = -\kappa_i \hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}}\,\mathrm{R}(\mathbf{\Delta}\boldsymbol{a})\mathbf{R}\,\hat{\mathbf{x}}_{\mathrm{n}i} - \beta_i \left[(\hat{\boldsymbol{\gamma}}_{1i}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\,\mathrm{R}(\mathbf{\Delta}\boldsymbol{a})^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}i})^2 - (\hat{\boldsymbol{\gamma}}_{2i}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\,\mathrm{R}(\mathbf{\Delta}\boldsymbol{a})^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}i})^2\right] \tag{6.12}$$

where $\mathbf{T} = [\mathbf{R},\, \mathbf{t}]$ is the current transformation estimate held prior to the registration phase. Following the minimization of (6.10), the current transformation estimate is updated as $\mathbf{T} = [\mathrm{R}(\mathbf{\Delta}\boldsymbol{a})\mathbf{R}, \mathrm{R}(\mathbf{\Delta}\boldsymbol{a})\mathbf{t} + \mathbf{\Delta t}]$.

The gradient equations required for minimizing (6.10) using the BFGS quasi-Newton solver are provided below, where $\nabla\mathbf{C}$ denotes the gradient of the objective function (6.10) with respect to the transformation parameters, $\boldsymbol{\Delta a}$ and $\boldsymbol{\Delta t}$, that are being optimized. In the equations that follow, the notation $\mathbf{J}_{a,b}$ signifies the Jacobian of a general expression, a, with respect to some variable, b.

$$\nabla\mathbf{C} = \sum_{i=1}^{n} \left(\nabla\mathbf{C}_{\mathrm{p}i} + \nabla\mathbf{C}_{\mathrm{n}i}\right) \tag{6.13}$$

$$\nabla\mathbf{C}_{\mathrm{p}i} = \left[\mathbf{J}_{C_{\mathrm{p}i},\mathbf{z}_i}\mathbf{J}_{\mathbf{z}_i,\boldsymbol{\Delta a}}, \quad \mathbf{J}_{C_{\mathrm{p}i},\mathbf{z}_i}\mathbf{J}_{\mathbf{z}_i,\boldsymbol{\Delta t}}\right]^{\mathsf{T}} \tag{6.14}$$

$$\mathbf{J}_{C_{\mathrm{p}i},\mathbf{z}_i} = \mathbf{z}_i^{\mathsf{T}}\mathbf{R}\boldsymbol{\Sigma}_i^{-1}\mathbf{R}^{\mathsf{T}} \tag{6.15}$$

$$\mathbf{J}_{\mathbf{z}_i,\boldsymbol{\Delta a}} = \left[\frac{\partial\mathbf{z}_i}{\partial\Delta a_x}, \frac{\partial\mathbf{z}_i}{\partial\Delta a_y}, \frac{\partial\mathbf{z}_i}{\partial\Delta a_z}\right] \tag{6.16}$$

$$\frac{\partial\mathbf{z}_i}{\partial\Delta a_j} = \frac{\partial\,\mathrm{R}(\boldsymbol{\Delta a})}{\partial\Delta a_j}^{\mathsf{T}}(\mathbf{y}_{\mathrm{p}i} - \boldsymbol{\Delta t}) \text{ for } j = \{x,\, y,\, z\} \tag{6.17}$$

$$\mathbf{J}_{\mathbf{z}_i,\boldsymbol{\Delta t}} = -\,\mathrm{R}(\boldsymbol{\Delta a})^{\mathsf{T}} \tag{6.18}$$

$$\nabla\mathbf{C}_{\mathrm{n}i} = \left[\mathbf{J}_{C_{\mathrm{n}i},\boldsymbol{\Delta a}}, \quad 0\right]^{\mathsf{T}} \tag{6.19}$$

$$\mathbf{J}_{C_{\mathrm{n}i},\boldsymbol{\Delta a}} = \left[\frac{\partial C_{\mathrm{n}i}}{\partial\Delta a_x}, \frac{\partial C_{\mathrm{n}i}}{\partial\Delta a_y}, \frac{\partial C_{\mathrm{n}i}}{\partial\Delta a_z}\right] \tag{6.20}$$

$$\begin{aligned}\frac{\partial C_{\mathrm{n}i}}{\partial\Delta a_j} = -\kappa\hat{\mathbf{y}}_{\mathrm{n}i}^{\mathsf{T}}\frac{\partial\,\mathrm{R}(\boldsymbol{\Delta a})}{\partial\Delta a_j}\mathbf{R}\hat{\mathbf{x}}_{\mathrm{n}i} - 2\beta\Bigg[&(\hat{\boldsymbol{\gamma}}_{1i}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\,\mathrm{R}(\boldsymbol{\Delta a})^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}i})(\hat{\boldsymbol{\gamma}}_{1i}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\frac{\partial\,\mathrm{R}(\boldsymbol{\Delta a})}{\partial\Delta a_j}^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}i}) \\ &-(\hat{\boldsymbol{\gamma}}_{2i}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\,\mathrm{R}(\boldsymbol{\Delta a})^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}i})(\hat{\boldsymbol{\gamma}}_{2i}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\frac{\partial\,\mathrm{R}(\boldsymbol{\Delta a})}{\partial\Delta a_j}^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}i})\Bigg] \text{ for } j = \{x,\, y,\, z\}\end{aligned} \tag{6.21}$$

In the foregoing equations, the expression $\frac{\partial\,\mathrm{R}(\boldsymbol{\Delta a})}{\partial\Delta a_j}$ signifies the 3×3 matrix of partial derivatives of $\mathrm{R}(\boldsymbol{\Delta a})$ with respect to a single element, Δa_j, of $\boldsymbol{\Delta a}$ for $j = \{x,\, y,\, z\}$.

The definitions for these partial derivatives are described in Appendix F.

6.5 Experimental Results and Discussion

In this section, the G-IMLOP algorithm is evaluated by registration experiments involving a mesh model of a human femur (Figures 6.2a) and 6.2b) created by segmentation from CT imaging. In addition, the IMLOP algorithm described in Chapter 5 and the ICP algorithm are also evaluated in order to compare their performance with G-IMLOP under anisotropic noise conditions.

Two experiments are conducted. The first experiment evaluates the algorithms under a wide range of anisotropic noise conditions. The second experiment simulates the clinical scenario of computer-assisted THR surgery by modeling measurements acquired by probing of the femur surface using a virtual pointer tracked by stereo vision.

In all studies, a common set of termination criteria is used for each evaluated algorithm, being when the magnitude of change in the estimated transformation, $\mathbf{T} = [\mathbf{R}, \mathbf{t}]$, falls below 0.001 mm and 0.001 degrees for two consecutive iterations or when 200 iterations are reached.

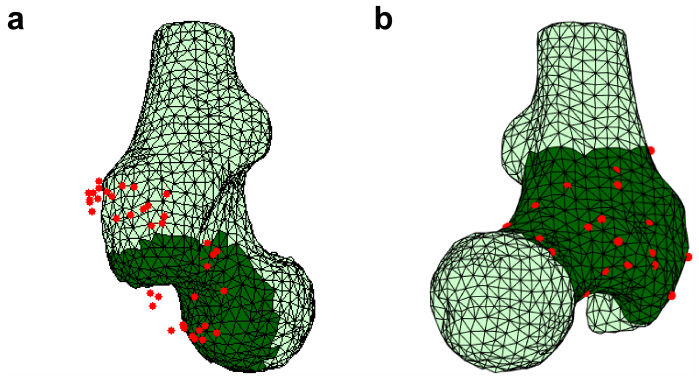

Figure 6.2: Femur mesh models used in (a) Experiment 1 showing a randomly generated and misaligned data shape and in (b) Experiment 2 showing a randomly generated data shape at the ground-truth alignment. Data shapes in each experiment were generated from the darkly shaded sub-region of each mesh.

6.5.1 Experiment 1: Registration with Anisotropic Noise

In this experiment, the anisotropic G-IMLOP and isotropic IMLOP algorithms are evaluated for registering data characterized by anisotropic noise, with registration performance also being compared to the ICP algorithm.

Table 6.1 lists the anisotropic noise models evaluated in this study, which include varying degrees of anisotropy in both the position and orientation data. The $\mathbf{\Sigma}$ column defines the square roots of the eigenvalues of the covariance matrix for positional (Gaussian) noise, while the κ and e columns are the concentration and eccentricity, respectively, of the orientation (Kent) noise. The *eccentricity*, e, takes values on the interval $[0, 1)$ and sets the value of the ellipticity parameter, β, as defined by

$$\beta = e\frac{\kappa}{2} \quad . \tag{6.22}$$

Note that $e = 0$ is the isotropic noise case, which reduces the Kent distribution to that of an isotropic Fisher distribution.

The noise models are arranged into six *test groups*; within each test group the eccentricities is set to 0.25, 0.5, and 0.75 while the other noise model parameters remain fixed. The test groups are arranged in order of increasing noise magnitude and anisotropy. Test groups 1–3 apply the same noise models as test groups 4–6, except that the standard deviations of noise are doubled. Note that a doubling of the orientation noise corresponds to a decrease in κ by a factor of 4, as understood by the following approximation relating the Fisher and the wrapped-Gaussian distributions[85]

$$\sigma^2_{\text{radians}} \approx \frac{1}{\kappa} \ . \tag{6.23}$$

Thus, the κ values of 3200 and 800 used in this study correspond to standard deviations of approximately 1 and 2 degrees, respectively, of overall orientation error.

As described in Chapter 5, each algorithm autonomously determines whether to accept or reject a registration outcome based on the final residual match errors. For ICP and IMLOP, this determination is made as described in Section 5.5.1 using match-error thresholds that are set to twice the effective standard deviation of the applied noise. For the position data, the effective standard deviation is defined as

$$\sigma^2_{\text{mm}} = (\lambda_1 \cdot \lambda_2 \cdot \lambda_3)^{1/3} \tag{6.24}$$

Table 6.1: Experiment 1: noise-model definitions, rejection-rate outcomes, and runtimes for the randomized registration study involving anisotropic noise.

				Rejection Rate (%)					
Test	**Noise Model**			**ICP**		**IMLOP**		**G-IMLOP**	
Group	**Σ**	κ	e	50	100	50	100	50	100
1	[0.5, 0.5, 0.25]	3200	0.25	0.0	0.0	5.3	1.0	4.0	1.0
	[0.5, 0.5, 0.25]	3200	0.5	0.0	0.0	5.3	1.0	4.0	0.7
	[0.5, 0.5, 0.25]	3200	0.75	0.0	0.0	13.3	2.3	5.0	0.3
2	[0.5, 0.5, 1]	3200	0.25	0.0	0.0	8.7	2.3	3.7	0.7
	[0.5, 0.5, 1]	3200	0.5	0.0	0.0	8.0	1.3	3.7	0.7
	[0.5, 0.5, 1]	3200	0.75	0.3	0.0	15.7	4.0	5.7	2.3
3	[0.25, 0.5, 1]	3200	0.25	0.0	0.0	8.0	0.7	5.3	1.0
	[0.25, 0.5, 1]	3200	0.5	0.0	0.0	8.0	0.0	4.3	1.0
	[0.25, 0.5, 1]	3200	0.75	0.7	0.0	18.3	2.7	6.3	2.7
4	[1, 1, 0.5]	800	0.25	0.0	0.0	7.7	0.7	4.0	0.7
	[1, 1, 0.5]	800	0.5	0.0	0.0	10.0	0.3	4.0	0.3
	[1, 1, 0.5]	800	0.75	0.0	0.0	14.0	2.3	4.7	0.3
5	[1, 1, 2]	800	0.25	0.0	0.0	7.0	2.3	2.3	0.3
	[1, 1, 2]	800	0.5	0.0	0.0	15.7	2.7	5.3	1.7
	[1, 1, 2]	800	0.75	0.0	0.0	18.0	1.0	5.7	0.7
6	[0.5, 1, 2]	800	0.25	0.0	0.0	11.0	2.7	4.3	1.0
	[0.5, 1, 2]	800	0.5	0.0	0.0	11.0	1.7	5.7	1.0
	[0.5, 1, 2]	800	0.75	0.0	0.0	17.3	3.0	9.0	1.7
			Runtime (sec.):	0.222	0.328	0.061	0.084	0.814	1.603
			Iterations:	139.1	145.0	34.6	33.1	27.8	27.2

This table defines the noise models used to generate data-shape noise for each test case. **Σ** defines the square roots of the eigenvalues of the covariance matrix for the positional (Gaussian) noise; κ and e define the concentration and eccentricity for the orientation (Kent) noise. Within each test group, the orientation eccentricity is varied from 0.25 to 0.75, while other parameters of the noise model remain fixed. Rejection rates for the registration outcomes are also listed for each algorithm for sample sizes of 50 and 100 samples. The average runtime and number of iterations are reported at the bottom of the table for each algorithm and sample size.

where λ_i are the eigenvalues of the covariance $\mathbf{\Sigma}$. For the orientation data, Equation (6.23) is used to define the standard deviation of noise.

For the G-IMLOP algorithm, the registration rejection criterion is a chi-square test performed on the sum of square Mahalanobis distances of the match residuals. This test is used because it accounts for the anisotropic elements of G-IMLOP's noise model. The square Mahalanobis distance normalizes each match residual by its variance along each dimension; thus, for positional data, this sum is distributed as a chi-square distribution with $3n$ degrees of freedom, where n is the number of samples.[66] A registration outcome is therefore rejected if this sum exceeds the value of a chi-square inverse cumulative density function with $3n$ degrees of freedom at a user-defined probability p

$$\text{Reject if} \quad \sum_{i=1}^{n} (\mathbf{y}_{\mathrm{p}i} - \mathbf{T}(\mathbf{x}_{\mathrm{p}i}))^{\mathsf{T}} R \Sigma_i^{-1} R^{\mathsf{T}} (\mathbf{y}_{\mathrm{p}i} - \mathbf{T}(\mathbf{x}_{\mathrm{p}i})) \geq \text{chi2inv}(p, 3n) \ . \qquad (6.25)$$

In similar manner, a chi-square test with $2n$ degrees of freedom is applied for the orientation data by converting the Kent noise model for the orientation match residuals into Gaussian form using the 2D wrapped-Gaussian approximation to the Kent distribution.[85] For the experiments in this chapter, a registration outcome for the G-IMLOP algorithm was rejected if either the position or orientation chi-square test failed at $p = 0.99$.

For each test case, 300 randomized registrations of the femur bone (Figure 6.2a)

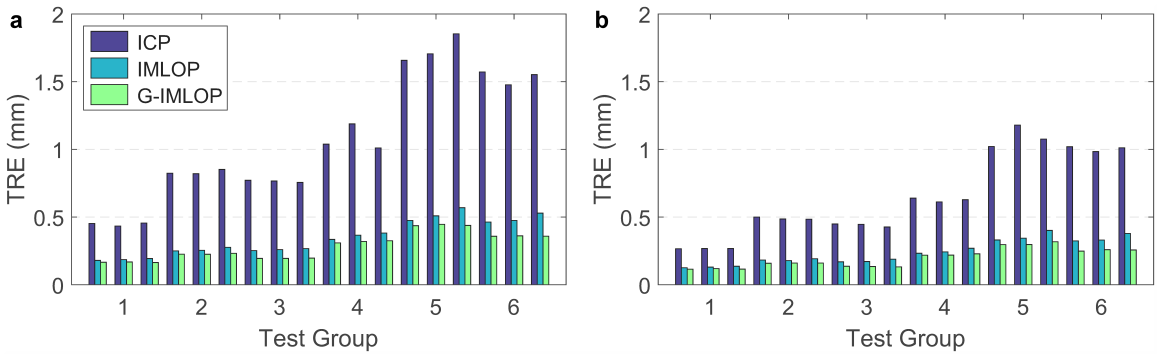

Figure 6.3: Experiment 1: anisotropic noise study. Average TREs of the accepted registration outcomes of G-IMLOP, IMLOP, and ICP are shown for sample sizes of (a) 50 and (b) 100 points. The test groups represent the different noise models used to generate data-shape noise, which are defined in Table 6.1. Note that each test group consists of three test cases, evaluating orientation noise eccentricities of 0.25, 0.5, and 0.75.

were conducted. Data-shapes were randomly generated from the darkly shaded region of the model, noised (using the generative noise models defined in Table 6.1), and then misaligned from the model before being registered back to the model using each algorithm. Random misalignments of 10 mm and 10 degrees were applied.

Figures 6.3a and 6.3b plot the average TREs of the accepted registration outcomes for each algorithm and test case, corresponding to data-shape sizes of 50 and 100 sample points, respectively. Rejection rates and runtime statistics are reported in Table 6.1.

As seen in Figure 6.3, both IMLOP and G-IMLOP widely outperform ICP in registration accuracy for all test cases, while G-IMLOP further outperforms IMLOP by significant margins. Within each test group, the accuracy advantage of G-IMLOP increases relative to the eccentricity of orientation noise, as expected. G-IMLOP also has lower rejection rates than IMLOP (Table 6.1), which indicates that for this study

the anisotropic noise model improves local minima avoidance in addition to improving accuracy at the correct alignment. As was the case in Chapter 5, ICP is generally unable to detect the inaccurate registration outcomes. Regarding runtime, IMLOP is approximately four times more efficient than ICP, due to faster convergence, whereas G-IMLOP is four times less efficient, due to the higher complexity of the anisotropic matching. PD-tree build time for the IMLOP and G-IMLOP algorithms was 10 ms (for a mesh size of 3130 triangles).

6.6 Experiment 2: Simulation of an Optically Tracked Pointer for THR Surgery

This experiment simulates the registration scenario of computer-assisted THR surgery by modeling an optically tracked pointer for taking surface measurements of the femur bone. The pointer geometry is assumed to be that of the Polaris® Passive Probe, Part Number 8700340 (NDI, Waterloo, ON, Canada). An optical tracker with a stereo vision measurement uncertainty of 0.25 mm and 0.125 mm standard deviation in the depth and depth-perpendicular directions, respectively, is assumed. It is further assumed that a force/torque sensor is fixed to the pointer and able to measure the surface-normal direction with an isotropic error of 0.2 degrees standard deviation.

Surface measurements were generated at random points from the darkly shaded region of the femur model shown in Figure 6.2b. For each sample, the pointer shaft

was oriented as near as possible to the surface-normal direction, subject to the optical spheres facing in the reverse tracker view direction (which was positioned lateral with respect to the femur). The pointer was further rotated, as needed, to limit the angle between the pointer shaft and the surface normal to 60 degrees maximum. Finally, a random perturbation uniformly chosen from [0,5] degrees was applied to the pointer orientation, and the pointer tip was then translated to touch the sample position.

Given a ground-truth pointer pose, measurements of the optically tracked spheres and of the surface-normal direction were simulated according to the stereo vision and force/torque sensor noise models. The final pointer-tip position and surface-normal direction were then computed by fitting the pointer geometry to the simulated tracked sphere positions using point-to-point registration.[32]

Noise models for the final pointer-tip position and surface-normal orientation were estimated for each pose via a Monte Carlo approach by generating 10,000 sets of tracked sphere and force/torque measurements and then fitting Gaussian and Kent distributions to the final values. Alternatively, the noise models could be computed analytically, using a technique such as propagation of uncertainty.[53]

Three-hundred randomized registration trials were conducted using data-shape sizes of 20 points and random data-shape misalignments of 5 mm and 5 degrees from ground truth. Figure 6.4 presents a histogram of the TRE outcomes for each algorithm, color-coded in blue and red to indicate trials where each algorithm autonomously determines the registration to be accurate (accepted trials) or inaccurate

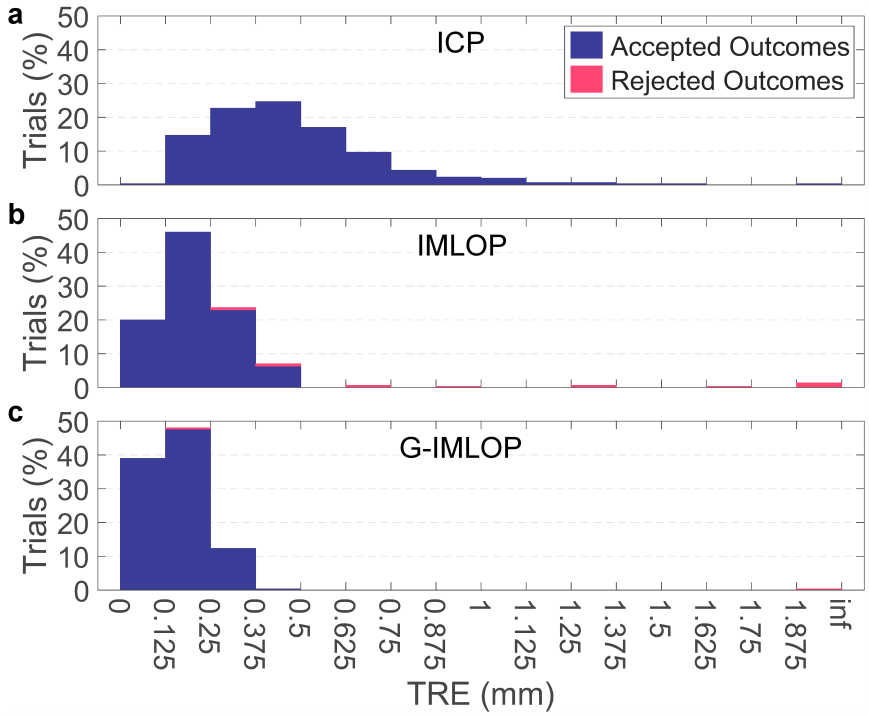

Figure 6.4: Experiment 2: tracked pointer simulation study. Histograms of the TREs for G-IMLOP, IMLOP, and ICP are shown; blue-coded (red-coded) values indicate accepted (rejected) trials, for which an algorithm has autonomously determined the registered shape alignment to be accurate (inaccurate).

(rejected trials), respectively. The average TREs of the accepted trials and the rejection rates are reported in Table 6.2. As seen in the figure and table, G-IMLOP and IMLOP achieve substantially lower registration error than ICP and, unlike ICP, robustly identify the inaccurate registration outcomes.

6.7 Concluding Remarks

The G-IMLOP algorithm extends the probabilistic framework introduced by the IMLOP algorithm for combining position and orientation information of the shapes

Table 6.2: Experiment 2: tracked pointer simulation study

	ICP	**IMLOP**	**G-IMLOP**
TRE (mm)	0.49	0.21	0.15
Rejection Rate (%)	0.0	4.7	0.7

This table reports the average TREs of the accepted registration outcomes and the registration rejection rates for each algorithm.

being registered by modeling anisotropic uncertainty in the position and orientation data. This extension provides significant improvement in registration accuracy over the isotropic IMLOP algorithm when registering oriented data characterized by anisotropic noise. In addition, G-IMLOP continues to offer a robust mechanism to autonomously characterize a registration outcome as likely to be accurate versus inaccurate. In the foregoing studies, G-IMLOP was found to not only improve registration accuracy relative to IMLOP's isotropic noise model, but also to increase how often a correct alignment was recovered from the registration outcome.

Importantly, an efficient implementation has been developed for the G-IMLOP, algorithm, including a fast PD-tree-based strategy for computing the most likely matches as defined by the G-IMLOP match error function, which accounts for both orientation and position data as well as anisotropic uncertainties. As demonstrated in the experiments of this chapter, the IMLOP algorithm computes a registration outcome four times faster on average than ICP, due to a faster rate-of-convergence and little added overhead per iteration, whereas the G-IMLOP algorithm is on average four times slower than ICP, due to its substantial increase in complexity resulting from the anisotropic noise models. Given the significant increase in the algorithmic

complexity of G-IMLOP relative to ICP, this runtime outcome is very satisfactory.

6.8 Contributions

The contributions from this chapter include:

4. Generalized IMLOP (G-IMLOP) Algorithm[27]

   (a) a probabilistic algorithm for registering *positional* and *orientational* feature data (i.e., oriented points) that are characterized by *anisotropic* uncertainty in the positions and/or orientations

   (b) an efficient implementation of the G-IMLOP algorithm consisting of:

      i. *G-IMLOP Correspondence Phase*: a fast PD-tree-based search for efficiently computing the most likely matches from a model shape considering both the position and orientation components of the match error, as well as the anisotropic uncertainty in these features

      ii. *G-IMLOP Registration Phase*: a gradient-based solution to the problem of registering corresponding sets of oriented points assuming anisotropic uncertainty in the positions and orientations; this solution is computed using the gradient equations of G-IMLOP's objective function, which is optimized using an off-the-shelf, nonlinear, BFGS quasi-Newton optimizer

(c) a mechanism for autonomously assessing a registration outcome in order to determine, with high confidence, whether a registration has succeeded or failed to compute an accurate shape alignment

6.9 Published Work

Material from this chapter has appeared in the following publication:

1. S. D. Billings and R. H. Taylor, "Generalized iterative most likely oriented-point (G-IMLOP) registration," *International Journal of Computer Assisted Radiology and Surgery*, vol. 10, no. 8, pp. 1213–1226, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11548-015-1221-2

Chapter 7

Projected Iterative Most Likely Oriented Point (P-IMLOP) Algorithm

This chapter describes the Projected Iterative Most Likely Oriented Point (P-IMLOP) algorithm, which is a special purpose algorithm for registering features segmented from tracked 2D ultrasound images. Thus, this chapter begins by motivating the novel elements of the P-IMLOP algorithm in light of the key challenges of registering tracked 2D ultrasound data. Content of this chapter has appeared in [29].

Consider the scenario of segmenting a surface contour from a tracked 2D ultrasound image. Although the position of each point on the surface contour can be fully described in 3D space based on the tracking data, the same cannot be said for the

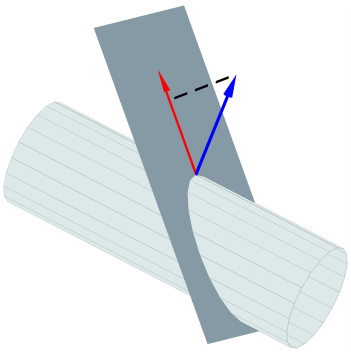

Figure 7.1: A geometric representation of a 2D ultrasound image plane intersecting a 3D object (cylinder) is shown. The object's surface normal as measured within ultrasound image plane (red) represents a projection of the true 3D surface normal (blue) onto the 2D image plane

3D orientations of the surface normal at each point on the surface contour. This is true because only the in-plane component of the true surface normal can be measured within the 2D ultrasound image. Any component of a surface normal that is oriented perpendicular to the ultrasound image cannot be measured. This concept is illustrated by the graphic of Figure 7.1, where the plane represents a 2D ultrasound image and the cylinder represents a 3D object being imaged by ultrasound. As demonstrated in the figure, the 3D surface normal of the cylinder must be projected to the ultrasound image plane in order to align the 3D surface normal with the normal of the surface contour that is measured within the ultrasound image. As a consequence of this fact, the P-IMLOP algorithm deals with aligning projections of 3D surface orientations onto the 2D image coordinates of the ultrasound images. Therefore, the P-IMLOP algorithm is designed to treat data point orientations as projected 2D data and to treat data point positions as 3D data.

For positional data, P-IMLOP incorporates an anisotropic noise model that en-

ables accurate modeling of the ultrasound imaging process, which has differing resolution in the axial, lateral, and elevational image directions (in order of decreasing resolution).[52] This results in anisotropic measurement uncertainties regarding the positions of imaged features. By modeling this anisotropic uncertainty, the P-IMLOP algorithm can improve registration accuracy.

In this chapter, *projection-oriented points* are defined as points having a 3D position associated with a 2D unit orientation vector defined on some arbitrary *image plane* in 3D space. In practice, these image planes define the orientations of the ultrasound images, whose orientations may be determined by using a tracking system, such as optical tracking, to track the pose of the ultrasound probe during image capture.

7.1 Clinical Perspective [1]

Development of the P-IMLOP algorithm is motivated by the clinical need to register surface contours in tracked ultrasound to preoperative CT-based anatomical surface models within the application of computer-assisted orthopedic surgery. Orthopedic joint replacement represents one of the earliest applications of robotics in surgery, dating back to the late 1980s[88] and early 1990s.[89,90] This was driven by the large market for total hip and knee replacement surgeries coupled with the technical advantages of working with bone, which are high contrast in X-ray or CT images

[1] Dr. Peter Kazanzides contributed to this section.

and structural rigidity. These factors enable robot systems to utilize preoperative images and models to plan the intervention, followed by a rigid registration between the preoperative (e.g., CT) and intraoperative (e.g., robot) coordinate systems. The rigidity of the bone ensures that the preoperative data remain valid and rigid fixation or tracking can be used to preserve registration accuracy during the procedure.

As an illustrative example, we consider the Robodoc® system (Think Surgical, Fremont CA), which was initially developed to machine the bone for the femoral component of the prosthesis in total hip replacement (THR) surgery (the conventional manual approach was used for the acetabular component). Early Robodoc procedures required the insertion of three metal screws (colloquially called *pins*) into the femur: one at the proximal end and two at the distal end (one medial and one lateral); later procedures eliminated one of the distal pins. The use of one or more distal pins stems from the requirement for accurate placement (within ± 1 mm) of the prosthesis – while the prosthesis head is located at the proximal end of the femur, the prosthesis stem is anchored in the femoral canal. Placing fiducials only at the proximal end of the femur would lead to larger errors (>1 mm) at the distal stem because distal positions would be more significantly affected by orientation error. While metal fiducials are easily segmented in the CT image and have features that can be precisely localized by the robot via a tactile search,[91] they introduce two major disadvantages: 1) an additional procedure is required to implant the pins, which adds to the cost and inconvenience, and 2) many patients reported significant knee pain as a result of

the distal pins, even if only one distal pin was used.[92] These disadvantages led to the development of a *pinless* registration method.[2] In this method, the surgeon or technician preoperatively segments specific regions of the proximal and distal femur to create a surface model. Intraoperatively, the surgeon collects multiple points by digitizing the bone surface in these regions and this "cloud" of points is registered to the preoperative surface model via an iterative optimization method. As with the fiducial-based method, it is necessary to segment and intraoperatively collect points on both the proximal and distal femur in order to achieve the clinically-required registration accuracy of ± 1 mm at all points along the prosthesis. But, intraoperatively collecting points by physically touching the bone surface leads to a conflict between high registration accuracy and reduced invasiveness. The existing Robodoc pinless procedure therefore relies on collecting 14 points on the exposed part of the proximal femur and 3 additional points on the distal femur. The proximal points are restricted to areas of the femur that are exposed by a normal incision and can be physically accessed; in practice, these points are on three "sides" of the proximal femur (which sides are accessible depends on the surgical approach). Also, points are not collected on the femoral head because it is typically resected prior to the registration procedure. The 3 distal points are collected by making small percutaneous incisions and inserting a thin probe to touch the bone surface.

Alternatively, several groups (e.g., [93–95]) have explored the use of 2D–3D registration between intraoperative X-ray images and preoperative CT. Although rea-

sonable accuracy may be achieved with these techniques, radiation exposure to the patient and operating room personnel and interference with the clinical workflow associated with the use of fluoroscopic C-arms remain issues.

Our goal is to use tracked ultrasound to improve the registration procedure by eliminating the need to invasively collect distal femur points. In the longer term, ultrasound may also be useful for collecting additional points on the proximal femur, especially to support even less invasive THR procedures[96] and possibly to further reduce the registration error.

The use of tracked ultrasound for bone registration is not a novel concept, as there is prior work in this area using both A-mode[97] and B-mode[18, 98–103] ultrasound. These prior works address significant technical problems that make it challenging to achieve accurate registration. First, the dimensional accuracy of ultrasound depends on the acoustic properties (e.g., speed of sound) of the tissue being imaged. Using a nominal value for the speed of sound, such as 1540 m/s, can lead to errors up to 5% in locating the bone surface; this can be improved by estimating the speed of sound as part of the registration process.[99] Second, it is challenging to segment the bone surface in ultrasound images; approaches to solve this problem include [18, 100, 101, 104, 105]. Finally, it is necessary to accurately calibrate the tracked ultrasound probe (i.e., to determine the transformation between the image frame and the tracked rigid body affixed to the probe). Ultrasound probe calibration is a general problem, with a large body of existing work outside the field of computer-assisted orthopedic surgery;

some of the above citations (e.g., [98]) also describe calibration methods in detail.

The contribution of this chapter is not in the above topics, but rather in the development and application of a new algorithm for performing the registration using a combination of ultrasound images and digitized bone surface points. This new algorithm, called P-IMLOP, is a modification of the G-IMLOP algorithm (Section 6) that was developed for registering position and orientation feature data. The key difference between the G-IMLOP and and the P-IMLOP algorithms is that projections of 3D orientations onto 2D image planes are registered as the orientation data rather than directly registering orientations in 3D, while the use of anisotropic noise models for the 3D data positions is retained. These characteristics are motivated by the problem of registering tracked 2D ultrasound data, as described above. Our experiments employ a recently developed active-echo method for calibration of the ultrasound tracking.[106, 107] In our phantom experiments, we manually segmented the ultrasound images and calibrated the speed of sound; the cited prior work in automatic segmentation[18, 100, 101, 104, 105] and speed-of-sound calibration[99] could be used to improve our implementation.

7.2 Probabilistic Model

The P-IMLOP algorithm incorporates a probabilistic framework formulated using 3D anisotropic Gaussian and 2D von Mises[85] distributions to model the uncertainty

of the position and projected orientation data, respectively.

Assuming independence of the positions and orientations, as well as zero-mean, independent uncertainty for each data point, then the *match likelihood function* for a projection-oriented data point, $\mathbf{x} = (\mathbf{x}_{3\mathrm{dp}}, \hat{\mathbf{x}}_{2\mathrm{dn}})$, that has been transformed by a current registration estimate, $[\mathbf{R},\, \mathbf{t}]$, is defined as

$$f_{\mathrm{match}}(\mathbf{x} \,|\, \mathbf{y}, \boldsymbol{\Sigma}, \kappa, \mathbf{R}_{\mathrm{pln}}, \mathbf{R}, \mathbf{t}) = \frac{1}{(2\pi)^{5/2}|\boldsymbol{\Sigma}|^{1/2}\, \mathrm{I}_0(\kappa)} \cdot e^{\kappa \frac{\mathrm{P}(\mathbf{R}_{\mathrm{pln}}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}})^{\mathsf{T}}}{\| \mathrm{P}(\mathbf{R}_{\mathrm{pln}}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}})\|} \hat{\mathbf{x}}_{2\mathrm{dn}} - \frac{1}{2}(\mathbf{y}_{3\mathrm{dp}} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{3\mathrm{dp}} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}} - \mathbf{t})} \tag{7.1}$$

where $\mathbf{y} = (\mathbf{y}_{3\mathrm{dp}}, \hat{\mathbf{y}}_{3\mathrm{dn}})$ is an oriented model-shape point that is assumed to be in correspondence with the projection-oriented data-shape point, $\mathbf{x} = (\mathbf{x}_{3\mathrm{dp}}, \hat{\mathbf{x}}_{2\mathrm{dn}})$, and where $\boldsymbol{\Sigma}$ is the *covariance matrix* of uncertainty in the measured 3D position, $\mathbf{x}_{\mathrm{p}}$, and κ is the *concentration parameter* of uncertainty in the measured 2D orientation vector, $\hat{\mathbf{x}}_{\mathrm{n}}$. In this formulation, $\mathbf{y}_{\mathrm{p}}$ is the *mean position* and $\mathrm{P}(\mathbf{R}_{\mathrm{pln}}^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}})^{\mathsf{T}} / \| \mathrm{P}(\mathbf{R}_{\mathrm{pln}}^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}})\|$ is the *mean direction* of the joint distribution. $\mathrm{P}(\cdot)$ is the projection operator that projects 3D orientations to the x-y plane by removing the z-coordinate. $\mathbf{R}_{\mathrm{pln}}$ is a rotation matrix defining the orientation of the image plane that contains the 2D orientation vector $\hat{\mathbf{x}}_{\mathrm{n}}$. Thus, $\mathbf{R}_{\mathrm{pln}}$ transforms a 2D orientation defined on the x-y plane (the local 2D coordinate system for measuring $\hat{\mathbf{x}}_{\mathrm{n}}$) to its in-plane orientation in 3D space. The value of $\mathbf{R}_{\mathrm{pln}}$ must be specified along with each projected orientation measurement, $\hat{\mathbf{x}}_{2\mathrm{dn}}$. In (7.1) the inverse of $\mathbf{R}_{\mathrm{pln}}$ is used to rotate an out-of-plane 3D

model orientation, $\hat{\mathbf{y}}_{3\mathrm{dn}}$, from global coordinates to the local image-plane coordinates, which is followed by a projection to the 2D image plane. Since projecting $\hat{\mathbf{y}}_{3\mathrm{dn}}$ to the image plance changes its length, this projection is renormalized in (7.1) in order to restore unit-length to the projected 2D orientation.

In the correspondence phase, a match for each projection-oriented data point is computed from the model by maximizing the match likelihood function of (7.1) over the set of all oriented model points. This operation reduces to computing the oriented model point, $\mathbf{y} = (\mathbf{y}_{3\mathrm{dp}}, \hat{\mathbf{y}}_{3\mathrm{dn}})$, that minimizes

$$\begin{aligned} &\mathrm{E}_{\text{P-IMLOP}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\Sigma}, \kappa, \mathbf{R}_{\mathrm{pln}}, \mathbf{R}, \mathbf{t}) = \\ &\frac{1}{2}(\mathbf{y}_{3\mathrm{dp}} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}} - \mathbf{t})^{\mathsf{T}}\mathbf{R}\boldsymbol{\Sigma}^{-1}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{3\mathrm{dp}} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}} - \mathbf{t}) - \kappa\frac{\mathrm{P}(\mathbf{R}_{\mathrm{pln}}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\mathrm{dn}})^{\mathsf{T}}}{\|\,\mathrm{P}(\mathbf{R}_{\mathrm{pln}}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\mathrm{dn}})\|}\hat{\mathbf{x}}_{2\mathrm{dn}} \end{aligned} \tag{7.2}$$

which is the *match error function* of the P-IMLOP algorithm.

In the registration phase, an updated pose for the data points is determined by computing the transformation, $\mathbf{T}$, that maximizes the total likelihood over all the matches, which simplifies to minimizing the following *total match error*

$$\begin{aligned} \mathbf{T} = \underset{[\mathbf{R},\,\mathbf{t}]}{\operatorname{argmin}} \Bigg(&\frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_{3\mathrm{dp}i} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}i} - \mathbf{t})^{\mathsf{T}}\mathbf{R}\boldsymbol{\Sigma}_i^{-1}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{3\mathrm{dp}i} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}i} - \mathbf{t}) \\ &- \sum_{i=1}^{n}\kappa_i\frac{\mathrm{P}(\mathbf{R}_{\mathrm{pln}i}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\mathrm{dn}i})^{\mathsf{T}}}{\|\,\mathrm{P}(\mathbf{R}_{\mathrm{pln}i}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\mathrm{dn}i})\|}\hat{\mathbf{x}}_{2\mathrm{dn}i}\Bigg)\,. \end{aligned} \tag{7.3}$$

7.3 Algorithm Overview

In the following discussion, a high-level overview of the P-IMLOP algorithm is presented, followed by sections detailing the two key sub-phases of P-IMLOP: the *correspondence phase* for computing the projection-oriented-point matches and the *registration phase* for computing the optimal alignment between these matches. A summary of the P-IMLOP algorithm is provided as Algorithm 7.1.

Algorithm 7.1. Projected Iterative Most Likely Oriented Point (P-IMLOP)

input : Data shape as a projection-oriented point cloud: $\mathbf{X} = \{(\mathbf{x}_{3\text{dp}i}, \hat{\mathbf{x}}_{2\text{dn}i})\}$
Model shape: Ψ
Orientations of the data image planes: $\{\mathbf{R}_{\text{pln}i}\}$
Data noise parameters: $\{\kappa_i\}$ and $\{\mathbf{\Sigma}_i\}$
Initial transformation: $\mathbf{T}_0 = [\mathbf{R}_0, \mathbf{t}_0]$

output: Final transformation, $\mathbf{T} = [\mathbf{R}, \mathbf{t}]$, that aligns $\mathbf{X}$ and Ψ

Initialize: $\mathbf{T} \leftarrow \mathbf{T}_0$

while *not converged* **do**

Compute the most likely point correspondences:

$$(\mathbf{y}_{3\text{dp}i}, \hat{\mathbf{y}}_{3\text{dn}i}) \leftarrow \underset{(\mathbf{y}_{3\text{dp}}, \hat{\mathbf{y}}_{3\text{dn}}) \in \Psi}{\operatorname{argmin}} \Bigg(\frac{1}{2}(\mathbf{y}_{3\text{dp}} - \mathbf{R}\mathbf{x}_{3\text{dp}i} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \mathbf{\Sigma}_i^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{3\text{dp}} - \mathbf{R}\mathbf{x}_{3\text{dp}i} - \mathbf{t}) - \kappa_i \frac{\mathrm{P}(\mathbf{R}_{\text{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{3\text{dn}})^{\mathsf{T}}}{\| \mathrm{P}(\mathbf{R}_{\text{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{3\text{dn}}) \|} \hat{\mathbf{x}}_{2\text{dn}i} \Bigg)$$

Register the point correspondences:

$$\mathbf{T} = \underset{[\mathbf{R}, \mathbf{t}]}{\operatorname{argmin}} \Bigg(\frac{1}{2} \sum_{i=1}^{n} (\mathbf{y}_{3\text{dp}i} - \mathbf{R}\,\mathbf{x}_{3\text{dp}i} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \mathbf{\Sigma}_i^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{3\text{dp}i} - \mathbf{R}\mathbf{x}_{3\text{dp}i} - \mathbf{t}) - \sum_{i=1}^{n} \kappa_i \frac{\mathrm{P}(\mathbf{R}_{\text{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{3\text{dn}i})^{\mathsf{T}}}{\| \mathrm{P}(\mathbf{R}_{\text{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{3\text{dn}i}) \|} \hat{\mathbf{x}}_{2\text{dn}i} \Bigg);$$

end

7.4 Correspondence Phase

This section describes an efficient implementation for the correspondence phase of the P-IMLOP algorithm, wherein the most likely oriented-point correspondence is computed from the model shape for each projection-oriented data point. An approach is described to efficiently compute the most likely oriented match that minimizes the match error function of (7.2). For the implementation described in this section, it is assumed that the following alternative match error function is used

$$\begin{aligned} \mathrm{E}_{\text{P-IMLOP}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\Sigma}, \kappa, \mathbf{R}_{\text{pln}}, \mathbf{R}, \mathbf{t}) = \\ \frac{1}{2}(\mathbf{y}_{3\text{dp}} - \mathbf{R}\mathbf{x}_{3\text{dp}} - \mathbf{t})^{\mathsf{T}}\mathbf{R}\boldsymbol{\Sigma}^{-1}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{3\text{dp}} - \mathbf{R}\mathbf{x}_{3\text{dp}} - \mathbf{t}) \\ + \kappa(1 - \frac{\mathrm{P}(\mathbf{R}_{\text{pln}}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\text{dn}})^{\mathsf{T}}}{\|\,\mathrm{P}(\mathbf{R}_{\text{pln}}^{\mathsf{T}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\text{dn}})\|}\hat{\mathbf{x}}_{2\text{dn}}) \end{aligned} \quad (7.4)$$

which, unlike (7.2), is always positive by addition of an extra κ term.

In Chapter 6 concerning the G-IMLOP algorithm, a method for computing a most likely match, given an anisotropic 3D data orientation and position, was presented based on a modified PD-tree search. The search strategy developed for the G-IMLOP algorithm cannot be directly applied to the P-IMLOP algorithm, however, because projections of the model surface normals to arbitrary image planes must be considered rather than the full 3D orientations. The approach to compute the most likely matches for the P-IMLOP algorithm combines ideas from the prior chapters with modification made for the projected orientations.

The PD tree for P-IMLOP is constructed around the model shape in the standard manner as described for positional feature data in Section 2.3. To illustrate P-IMLOP's PD-tree search procedure, consider that given a projection-oriented data point, $(\mathbf{x}_{3\mathrm{dp}}, \hat{\mathbf{x}}_{2\mathrm{dn}})$, with known image plane orientation, $\mathbf{R}_{\mathrm{pln}}$, and noise-model parameters, $\mathbf{\Sigma}$ and κ, the task is to determine at each node whether any oriented model point contained therein can possibly produce a lower match error than the current candidate for the best match. Since the PD-tree node boundaries are based on the datum positions, what is required is an upper bound ($E_{\mathrm{p,max}}$) on the positional component (E_{p}) of the match error

$$E_{\mathrm{p}} = \frac{1}{2}(\mathbf{y}_{3\mathrm{dp}} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}} - \mathbf{t})^{\mathsf{T}}\mathbf{R}\mathbf{\Sigma}^{-1}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{3\mathrm{dp}} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}} - \mathbf{t}) \tag{7.5}$$

beyond which any match candidate may be ruled out. This bound is obtained by subtracting a lower bound ($E_{\mathrm{n,min}}$) on the orientational component of match error from the total match error of the current candidate for the best match (E_{best})

$$E_{\mathrm{p,max}} = E_{\mathrm{best}} - E_{\mathrm{n,min}}\,. \tag{7.6}$$

In Chapter 5 a method is described for lower-bounding the orientation error of all datums within the node using the average orientation and maximum angular deviation from the average orientation. For P-IMLOP, the matter of projected orientations complicates the issue. We therefore take the straightforward approach of assuming

perfect orientation alignment of all match candidates within a node when ruling out nodes to be searched. Using the match error equation of (7.4), then under the assumption of perfect orientation alignment $E_{\text{p,max}}$ becomes simply equal to E_{best}.

For an isotropic noise model, nodes would be ruled out by simply checking that the distance of $\mathbf{R}\mathbf{x}_{\text{3dp}} + \mathbf{t}$ from the node boundary is greater than the maximum search distance inferred from $E_{\text{p,max}}$. For the anisotropic noise model of P-IMLOP, the level sets of the positional match error component of (7.5) form ellipsoids centered at $\mathbf{R}\mathbf{x}_{\text{3dp}}+\mathbf{t}$ rather than spheres. The strategy then is to compute the ellipsoid boundary corresponding to the level-set value inferred from $E_{\text{p,max}}$ and to check if this ellipsoid intersects the oriented bounding box of the node, as previously described in Chapters 4 and 6. If no intersection exists, then the node is ruled out of the search. The ellipsoid so described is defined from the following inequality

$$(\mathbf{y}_{\text{3dp}} - \mathbf{R}\mathbf{x}_{\text{3dp}} - \mathbf{t})^{\mathsf{T}}\mathbf{R}\boldsymbol{\Sigma}^{-1}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{\text{3dp}} - \mathbf{R}\mathbf{x}_{\text{3dp}} - \mathbf{t}) \leq 2E_{\text{p,max}}\,. \tag{7.7}$$

An efficient technique for computing the intersection test between an ellipsoid and the oriented bounding box of the node is described by Larsson.[67]

A summary of the PD-tree node-search strategy for P-IMLOP is provided as Algorithm 7.2. The general form of $E_{\text{p,max}}$ is shown in Algorithm 7.2 with an unspecified value for $E_{\text{n,min}}$, since non-zero values of $E_{\text{n,min}}$ could be computed for a tighter and more efficient bound, although a method for doing so is not addressed in this work.

Note that for a model shape represented by a mesh (rather than by a point cloud) the datum point, $\mathbf{y}_j = (\mathbf{y}_{\mathrm{p}j}, \hat{\mathbf{y}}_{\mathrm{n}j})$, that is used to compute the datum match error in line 6 of Algorithm 7.2 is obtained by setting $\hat{\mathbf{y}}_{\mathrm{n}j}$ to the triangle normal and $\mathbf{y}_{\mathrm{p}j}$ to the point on the datum triangle that is the most likely match by position from $\mathbf{x}_{\mathrm{p}}$, which is discussed in Appendix B.

Algorithm 7.2. PD-Tree Node Search for Projection-Oriented-Point Matching

input : Projection-oriented data point to be matched: $\mathbf{x} = (\mathbf{x}_{3\mathrm{dp}}, \hat{\mathbf{x}}_{2\mathrm{dn}})$
Data noise-model parameters: $\mathbf{\Sigma}$ and κ
Orientation of the data image plane: $\mathbf{R}_{\mathrm{pln}}$
Current registration parameters: $[\mathbf{R}, \mathbf{t}]$
Current candidate for best match and its match error: $[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}]$
Node being searched: $\mathcal{N}$

output: Updated best match: $[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}]$

1 Compute an upper bound on the positional component of match error for this node:

$$E_{\mathrm{p,max}} = E_{\mathrm{best}} - E_{\mathrm{n,min}}$$

2 Compute an intersection test between the ellipsoid

$$\mathcal{E} = \{\mathbf{z} \mid (\mathbf{z} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}} - \mathbf{t})^{\mathsf{T}}\mathbf{R}\mathbf{\Sigma}^{-1}\mathbf{R}^{\mathsf{T}}(\mathbf{z} - \mathbf{R}\mathbf{x}_{3\mathrm{dp}} - \mathbf{t}) \leq 2E_{\mathrm{p,max}}\}$$

and the oriented bounding box of the node, $\mathcal{N}.OBB$

3 **if** $\mathcal{E}$ *intersects* $\mathcal{N}.OBB$ **then**
4 **if** $\mathcal{N}$ *is a leaf node* **then**
5 **foreach** $datum_j \in \mathcal{N}$ **do**
6 Compute the match error for this datum:
$E_j \leftarrow \mathrm{E}_{\text{P-IMLOP}}(\mathbf{x}, \mathbf{y}_j, \Sigma, \kappa, \mathbf{R}_{\mathrm{pln}}, \mathbf{R}, \mathbf{t})$ (Equ. (7.4), Appendix B)
7 **if** $E_j < E_{best}$ **then** update the best match:
$[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}] \leftarrow [\mathbf{y}_j, E_j]$
8 **end**
9 **else**
10 Search the left and right child nodes of $\mathcal{N}$
11 **end**
12 **end**
13 Return the updated best match within this node: $[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}]$

7.5 Registration Phase

This sub-section describes an implementation for the registration phase of the P-IMLOP algorithm, wherein the projection-oriented data points $\{(\mathbf{x}_{3\text{dp}i}, \hat{\mathbf{x}}_{2\text{dn}i})\}$ are registered with a matching set of oriented model points $\{(\mathbf{y}_{3\text{dp}i}, \hat{\mathbf{y}}_{3\text{dn}i})\}$ computed from a prior correspondence phase. The goal is to compute the transformation, $\mathbf{T} = [\mathbf{R},\, \mathbf{t}]$, that maximizes the total match likelihood, which simplifies to minimizing the total match error of Equation (7.3), which is repeated below for ease of reference

$$\mathbf{T} = \underset{[\mathbf{R},\,\mathbf{t}]}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n} (\mathbf{y}_{3\text{dp}i} - \mathbf{R}\,\mathbf{x}_{3\text{dp}i} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_i^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{3\text{dp}i} - \mathbf{R}\mathbf{x}_{3\text{dp}i} - \mathbf{t}) \right.$$
$$\left. - \sum_{i=1}^{n} \kappa_i \frac{\mathrm{P}(\mathbf{R}_{\text{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{3\text{dn}i})^{\mathsf{T}}}{\| \mathrm{P}(\mathbf{R}_{\text{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{3\text{dn}i}) \|} \hat{\mathbf{x}}_{2\text{dn}i} \right).$$

Due to the non-linearity of this objective function, iterative optimizaiton methods are required. As in the prior chapters, we employ the gradient-based BFGS quasi-Newton optimization method of the dlib open-source C++ library.[87]

In order to minimize the objective of (7.3) using the unconstrained quasi-Newton optimizer, a re-parameterization of the rotation matrix, $\mathbf{R}$, is required in order to enforce the constraints for valid rotation during optimization, namely $\mathbf{R}^{\mathsf{T}}\mathbf{R} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$. Due to its intuitive appeal and natural pairing to the 3 degrees of rotational freedom, the Rodrigues parameterization is used, which represents rotation as a 3-vector $\boldsymbol{a} = [a_x,\, a_y,\, a_z]^{\mathsf{T}}$ whose directionality and magnitude signify the axis

and angular extent of the rotation, respectively. The notation $\mathrm{R}(\boldsymbol{a})$ is used to signify the 3×3 rotation matrix corresponding to Rodrigues vector $\boldsymbol{a}$, which was previously defined in Equation (6.7).

In each registration phase, a new transformation is computed as an incremental update ($\boldsymbol{\Delta}\mathbf{T}$) on the prior transformation. Thus, given a current estimate of the transformation, $\mathbf{T} = [\mathbf{R},\, \mathbf{t}]$, the objective function to be minimized is

$$\boldsymbol{\Delta}\mathbf{T} = \underset{[\boldsymbol{\Delta a}, \boldsymbol{\Delta t}]}{\operatorname{argmin}} \sum_{i=1}^{n} (C_{\mathrm{p}i} + C_{\mathrm{n}i}) \tag{7.8}$$

$$C_{\mathrm{p}i} = \frac{1}{2} \mathbf{z}_i^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_i^{-1} \mathbf{R}^{\mathsf{T}} \mathbf{z}_i \tag{7.9}$$

$$\mathbf{z}_i = \mathrm{R}(\boldsymbol{\Delta a})^{\mathsf{T}} (\mathbf{y}_{\mathrm{p}i} - \mathrm{R}(\boldsymbol{\Delta a})(\mathbf{R}\mathbf{x}_{\mathrm{p}i} + \mathbf{t}) - \boldsymbol{\Delta t}) \tag{7.10}$$

$$C_{\mathrm{n}i} = -\kappa_i \frac{\mathrm{P}(\mathbf{R}_{\mathrm{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}}\, \mathrm{R}(\boldsymbol{\Delta a})^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}i})^{\mathsf{T}}}{\| \,\mathrm{P}(\mathbf{R}_{\mathrm{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}}\, \mathrm{R}(\boldsymbol{\Delta a})^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}i})\|} \hat{\mathbf{x}}_{\mathrm{n}i}\,. \tag{7.11}$$

Subsequent to this minimization, the current estimate of the transformation is updated as $\mathbf{T} = [\mathrm{R}(\boldsymbol{\Delta a})\mathbf{R}, \mathrm{R}(\boldsymbol{\Delta a})\mathbf{t} + \boldsymbol{\Delta t}]$.

The quasi-Newton optimizer requires the gradient ($\nabla\mathbf{C}$) of the objective function of (7.8) with respect to the transformation parameters $\boldsymbol{\Delta a}$ and $\boldsymbol{\Delta t}$. These gradient equations are provided below, where the notation $\mathbf{J}_{a,b}$ signifies the Jacobian of an

expression a with respect to some variable b.

$$\nabla \mathbf{C} = \sum_{i=1}^{n} (\nabla \mathbf{C}_{\mathrm{p}i} + \nabla \mathbf{C}_{\mathrm{n}i}) \tag{7.12}$$

$$\nabla \mathbf{C}_{\mathrm{p}i} = \left[\mathbf{J}_{C_{\mathrm{p}i},\mathbf{z}_i} \mathbf{J}_{\mathbf{z}_i,\boldsymbol{\Delta a}} \,, \quad \mathbf{J}_{C_{\mathrm{p}i},\mathbf{z}_i} \mathbf{J}_{\mathbf{z}_i,\boldsymbol{\Delta t}} \right]^{\mathsf{T}} \tag{7.13}$$

$$\mathbf{J}_{C_{\mathrm{p}i},\mathbf{z}_i} = \mathbf{z}_i^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_i^{-1} \mathbf{R}^{\mathsf{T}} \tag{7.14}$$

$$\begin{aligned} \mathbf{J}_{\mathbf{z}_i,\boldsymbol{\Delta a}} = - \Big[& \tfrac{\partial \mathrm{R}(\boldsymbol{\Delta a})}{\partial \Delta a_x}^{\mathsf{T}} (\mathbf{y}_{3\mathrm{dp}i} - \boldsymbol{\Delta t}), \quad \tfrac{\partial \mathrm{R}(\boldsymbol{\Delta a})}{\partial \Delta a_y}^{\mathsf{T}} (\mathbf{y}_{3\mathrm{dp}i} - \boldsymbol{\Delta t}), \\ & \tfrac{\partial \mathrm{R}(\boldsymbol{\Delta a})}{\partial \Delta a_z}^{\mathsf{T}} (\mathbf{y}_{3\mathrm{dp}i} - \boldsymbol{\Delta t}) \Big] \end{aligned} \tag{7.15}$$

$$\mathbf{J}_{\mathbf{z}_i,\boldsymbol{\Delta t}} = -\,\mathrm{R}(\boldsymbol{\Delta a})^{\mathsf{T}} \tag{7.16}$$

$$\nabla \mathbf{C}_{\mathrm{n}i} = \left[\mathbf{J}_{C_{\mathrm{n}i},\mathbf{y}_{\mathrm{prj}i}} \mathbf{J}_{\mathbf{y}_{\mathrm{prj}i},\boldsymbol{\Delta a}}, \quad 0 \right]^{\mathsf{T}} \tag{7.17}$$

$$\mathbf{y}_{\mathrm{prj}i} = \mathrm{P}(\mathbf{R}_{\mathrm{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}}\, \mathrm{R}(\boldsymbol{\Delta a})^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}i}) \tag{7.18}$$

$$\mathbf{J}_{C_{\mathrm{n}i},\mathbf{y}_{\mathrm{prj}i}} = -\kappa_i \hat{\mathbf{x}}_{2\mathrm{dn}i}^{\mathsf{T}} \left(\frac{1}{\|\mathbf{y}_{\mathrm{prj}i}\|_2} \mathbf{I} - \frac{\mathbf{y}_{\mathrm{prj}i} \mathbf{y}_{\mathrm{prj}i}^{\mathsf{T}}}{\|\mathbf{y}_{\mathrm{prj}i}\|_2^3} \right) \tag{7.19}$$

$$\begin{aligned} \mathbf{J}_{\mathbf{y}_{\mathrm{prj}i},\boldsymbol{\Delta a}} = \Bigg[& \mathrm{P}(\mathbf{R}_{\mathrm{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \frac{\partial \mathrm{R}(\boldsymbol{\Delta a})}{\partial \Delta a_x}^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}i}),\ \mathrm{P}(\mathbf{R}_{\mathrm{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \frac{\partial \mathrm{R}(\boldsymbol{\Delta a})}{\partial \Delta a_y}^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}i}), \\ & \mathrm{P}(\mathbf{R}_{\mathrm{pln}i}^{\mathsf{T}} \mathbf{R}^{\mathsf{T}} \frac{\partial \mathrm{R}(\boldsymbol{\Delta a})}{\partial \Delta a_z}^{\mathsf{T}} \hat{\mathbf{y}}_{3\mathrm{dn}i}) \Bigg] \end{aligned} \tag{7.20}$$

In the foregoing equations, $\frac{\partial \mathrm{R}(\boldsymbol{\Delta a})}{\partial \Delta a_x}$ signifies the 3×3 matrix of partial derivatives of $\mathrm{R}(\boldsymbol{\Delta a})$ with respect to element Δa_x of $\boldsymbol{\Delta a} = [\Delta a_x,\, \Delta a_y,\, \Delta a_z]^{\mathsf{T}}$, and so on for $\frac{\partial \mathrm{R}(\boldsymbol{\Delta a})}{\partial \Delta a_y}$ and $\frac{\partial \mathrm{R}(\boldsymbol{\Delta a})}{\partial \Delta a_z}$. The definitions for these partial derivatives are described in Appendix F.

7.6 Experimental Results and Discussion

In this section, the performance of the P-IMLOP algorithm is evaluated for registering a femur bone in application to computer-assisted total hip replacement (THR) surgery. To perform this evaluation, a phantom study was performed.

A bone- and tissue-mimicking phantom was constructed by encasing a plastic Sawbones femur (Item #1106, Pacific Research Laboratories, Inc., Vashon Island WA) within a plastisol (Item #2228LP, M-F Mfg., Fort Worth TX) casting formed from a leg-shaped mold (Figure 7.2). Optically tracked markers were rigidly affixed to the proximal and distal ends of the femur serving the dual purposes of establishing a patient coordinate system (i.e., a reference frame) for instrument tracking and of establishing a ground-truth alignment for the registration. The model shape used for the femur registration study was a mesh of the femur surface created by segmenting a CT scan of the phantom using intensity-based thresholding followed by manual fix-up in order to obtain an accurate segmentation of the femur surface. The ground-truth transformation between patient and CT coordinates was determined by manually segmenting the center positions of the optically tracked fiducials from the CT image and performing a point-to-point registration with the positions of the fiducials in patient coordinates as measured by the optical tracker (fusionTrack 500, Atracsys, Puidoux Switzerland).

Experimental data was acquired in patient-based coordinates by sampling points from the femur surface via optical tracking of both a pointer and an ultrasound

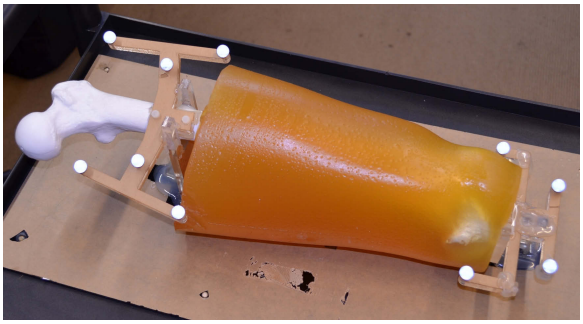

Figure 7.2: Femur phantom constructed to assess registration accuracy in application to total hip replacement surgery.

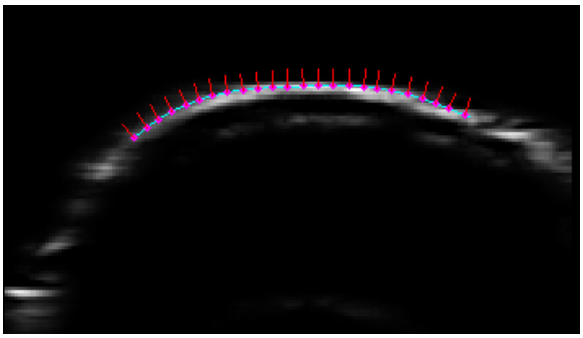

Figure 7.3: An example manual segmentation of the femur surface from an ultrasound image. The segmentation was performed by fitting a smoothing spline (shown as a light blue line) to points manually positioned along the bone contour. The smoothing spline was then sampled at 1 mm intervals to obtain both positions and normal orientations along the contour.

probe. Surface features were obtained from ultrasound by manually fitting a cubic smoothing spline to the bone surface in each ultrasound image and then sampling points from the spline at 1 mm intervals along with the affiliated 2D (image-plane) surface orientations (Figure 7.3).

Six experimental datasets were acquired, each including a unique set of 14 points sampled by a tracked pointer from the proximal femur at locations as specified by the Robodoc protocol[108] for the anterolateral and direct lateral approaches to THR surgery; for these approaches, the anterior surface of the proximal femur is accessible.

Depending on the registration method being tested, each dataset additionally included a unique set of 3 points sampled by the tracked pointer at the far distal end of the femur shaft near the knee via lateral, medial, and anterior incisions or surface features from 30 tracked ultrasound images spanning the lower (distal) half of the shaft. Each set of 30 ultrasound images consisted of 10 images captured from lateral, medial, and anterior scans of the femur shaft with the probe's elevational axis oriented in the patient-superior direction. Three sets of 30 ultrasound images were acquired; thus, each image set was used in two of the six datasets (paired as [1,4], [2,5], and [3,6]).

For the registration studies, the experimental datasets were randomly misaligned from the known ground-truth alignment and then registered back to the femur model. Three registration methods were assessed: 1) the ICP algorithm with pointer samples of the proximal and distal femur, 2) the ICP algorithm with pointer samples of the proximal femur and ultrasound samples of the distal femur, and 3) the P-IMLOP algorithm using the same data as the second ICP method.

For the P-IMLOP noise model, points segmented from ultrasound were assigned orientation concentrations of $\kappa = 100$ (angular standard deviation ≈ 5.7 degrees) and positional covariances with standard deviations of 1 mm in-plane and 1.5 mm out-of-plane with respect to the ultrasound image orientation. Although the in-plane axial resolution is higher than the in-plane lateral resolution, we use a common variance for both in-plane directions due to added axial uncertainty from speed-of-sound. Points sampled by tracked pointer were assigned isotropic covariances of

1 mm standard deviation; the orientation component was effectively disabled for these points by setting $\kappa = 0$. For the segmented ultrasound data, optimal values for the comparatively greater out-of-plane variance and for the orientation concentration, κ, were experimentally determined by testing a range of values. The results of this testing are presented later in this section following the primary experimental results. Note that the orientation concentration value used here is different than the value used in our prior publication of this work,[29] where a concentration of $\kappa = 50$ was used rather than $\kappa = 100$. The higher orientation concentration value is used here because it was later found to provide further improvement in registration accuracy.

A multi-series study was performed to evaluate registration performance at various magnitudes of initial misalignment including 3 mm (3 degrees), 5 mm (5 degrees), 5 mm (10 degrees), and 5 mm (20 degrees). For each dataset and misalignment magnitude, 100 registration trials were performed, each generating a different random misalignment. The same data and misalignments were applied to each registration method. For this application, the larger misalignments actually have little relevance, since close initialization is possible by first registering only the 14 points that were sampled from the proximal femur via the tracked pointer. This is especially true since these points are acquired via the guided Robodoc protocol and thus, from the outset, have approximately known correspondence regions on the model. Nonetheless, the larger misalignments are included in this study in the interest of further investigating the performance of each registration method.

The accuracy of each registration outcome relative to the ground-truth transformation was evaluated using a TRE measured at three different points located approximately central to the femur canal, including near the femur head and at displacements of 130 mm and 260 mm down the shaft. TRE values were calculated from the subset of trials that registered successfully. Registration "success" was determined automatically by each algorithm by using an upper threshold on the average residual match error at termination. For this application, residuals for the proximally sampled data were evaluated independently from the distally sampled data, requiring that both pass. The thresholds for position and orientation match error were set to 1 mm and 4 degrees, respectively, with the orientation condition applying only to the P-IMLOP method.

Table 7.1 reports the outcomes of this study. Average results are reported for each of the 6 experimental datasets independently (first six rows) as well as aggregately (seventh row) for each algorithm. Since the TREs of the successful trials were very similar across all misalignments, only one set of TRE values is reported, being the averages over the entire study. Differences in the registration failure rates were significant, however, and are reported independently for each misalignment magnitude (denoted by the coding A–D in order of increasing magnitude of misalignment). The overall averages for the runtime and the number of algorithmic iterations of the successful trials are also listed for each algorithm. Figures 7.4, 7.5, and 7.6 visually characterize the average distribution of TRE over the entire model shape for each

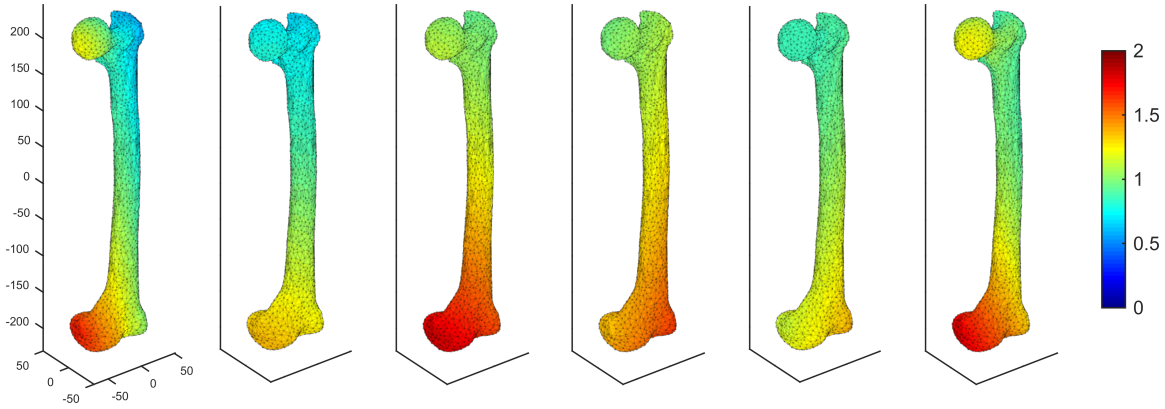

Figure 7.4: TREs of the ICP algorithm with distal incision data. Average TRE (in mm) at all points on the femur surface is shown for the ICP algorithm applied while using the distal incision data. The TRE is averaged over all successfully registered randomized trials and over all magnitudes of misalignment. Each image depicts results for one of the 6 experimental datasets

experimental dataset, with each figure representing the results for one algorithm.

As seen in the table and figures, the P-IMLOP algorithm applied to tracked ultrasound data achieves the best result, showing significantly improved registration accuracy compared to the ICP algorithm applied to the same data. Both tracked ultrasound methods have substantially higher registration accuracy than ICP with distal incision data. A further advantage for P-IMLOP is a much lower registration failure rate than either of the ICP methods. A disadvantage of P-IMLOP is that its runtime is approximately twice that of ICP using similar data, although P-IMLOP converges in half the iterations.

As a matter of investigation, registrations were also conducted using the IMLOP algorithm (Chapter 5) applied to tracked ultrasound data by treating the 2D orientations from ultrasound as 3D orientations defined to be in-plane with the ultrasound image (which is a poor assumption in general, recalling Figure 7.1). The result had

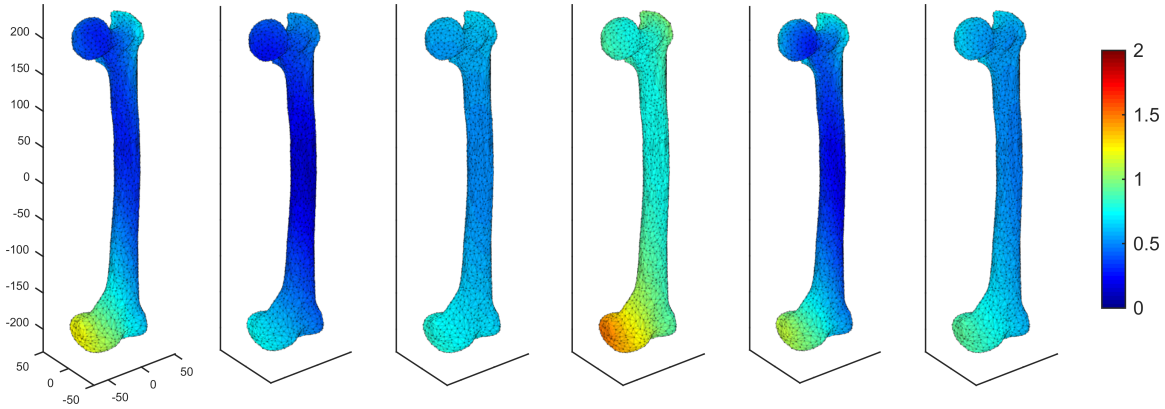

Figure 7.5: TREs of the ICP algorithm with tracked ultrasound data. Average TRE (in mm) at all points on the femur surface is shown for the ICP algorithm applied while using the tracked ultrasound data. The TRE is averaged over all successfully registered randomized trials and over all magnitudes of misalignment. Each image depicts results for one of the 6 experimental datasets

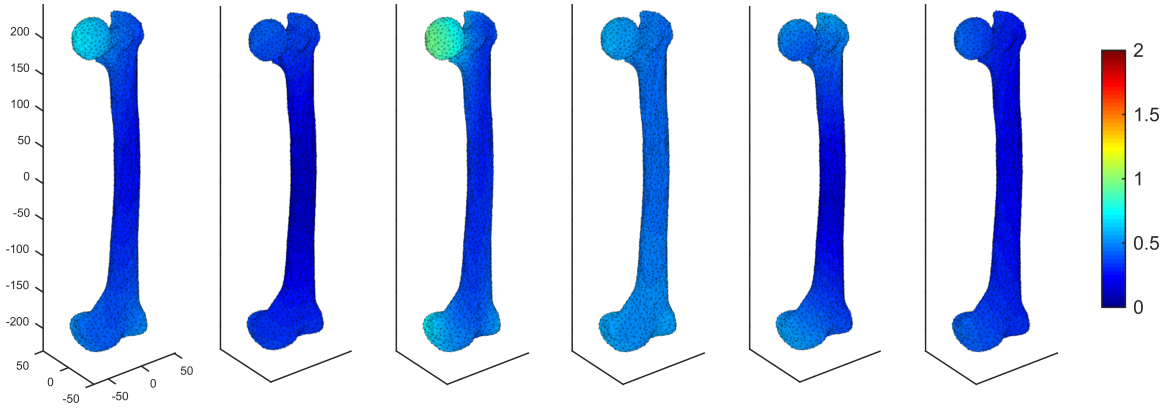

Figure 7.6: TREs of the P-IMLOP algorithm with tracked ultrasound data. Average TRE (in mm) at all points on the femur surface is shown for the P-IMLOP algorithm applied while using the tracked ultrasound data. The TRE is averaged over all successfully registered randomized trials and over all magnitudes of misalignment. Each image depicts results for one of the 6 experimental datasets

Table 7.1: Femur registration study outcomes.

	TRE (mm)			Failure Rate (%)					
Algorithm	Head	Mid	Distal	A	B	C	D	Iter.	Runtime
ICP with Distal Incision Data	0.69	0.80	0.97	7	17	14	10	35	0.055
	0.72	0.85	1.04	2	12	8	7	32	0.053
	0.99	1.12	1.33	5	9	10	7	31	0.050
	1.06	1.19	1.35	3	6	10	9	34	0.054
	0.89	1.00	1.13	0	8	8	6	33	0.051
	0.92	0.92	1.12	2	5	8	6	35	0.053
Average:	0.88	0.98	1.16	3.2	9.5	9.7	7.5	33	0.053
ICP with Tracked Ultrasound Data	0.54	0.33	0.40	0	5	4	0	232	1.207
	0.45	0.23	0.25	3	13	19	16	230	1.133
	0.62	0.53	0.56	0	2	10	10	247	1.168
	0.86	0.77	0.81	0	7	15	22	222	1.017
	0.60	0.35	0.31	0	1	0	1	252	1.146
	0.59	0.52	0.54	0	2	9	13	244	1.084
Average:	0.61	0.45	0.47	0.5	5.0	9.5	10.3	238	1.128
P-IMLOP with Tracked Ultrasound Data	0.41	0.29	0.33	0	0	0	0	113	2.425
	0.34	0.17	0.16	0	0	1	4	96	2.204
	0.43	0.32	0.31	0	0	0	6	114	2.379
	0.48	0.46	0.48	1	3	20	20	96	2.024
	0.49	0.27	0.20	0	0	0	1	102	2.256
	0.29	0.23	0.24	0	0	12	13	117	2.325
Average:	0.41	0.29	0.28	0.2	0.5	5.5	7.3	106	2.273

Results are shown for the proposed P-IMLOP algorithm applied to tracked ultrasound data and for the ICP algorithm applied to both distal incision and tracked ultrasound data. Registrations were conducted by randomly misaligning experimental data from the ground-truth alignment. 100 randomized trials were performed for each of 4 different misalignment magnitudes and each of 6 different datasets. Average results are reported for the datasets independently (first six rows of each algorithm) and for all datasets combined (seventh row of each algorithm). TRE, runtime, and number of iterations are averages of the successfully registered trials across all magnitudes of misalignment, with TRE measured near the head of the femur and at displacements of 130 mm (Mid) and 260 mm (Distal) down the shaft. Registration failure rates are reported individually for each misalignment magnitude, including (A) 3 mm (3 degrees), (B) 5 mm (5 degrees), (C) 5 mm (10 degrees), and (D) 5 mm (20 degrees).

a virtually identical TRE but lower registration failure rate than the ICP algorithm applied to the same tracked ultrasound data.

The results of the experiments referred to earlier for determining the optimal settings for the orientation concentration and for the out-of-plane variance parameters of the P-IMLOP algorithm (for the features segmented from ultrasound) are presented in Figures 7.7 and 7.8, respectively. These figures show the TRE outcomes (averaged over all six datasets) that were obtained by running the registration study using a range of settings for each parameter. As seen in Figure 7.7, including the orientation data in the registration significantly improves registration accuracy compared to disregarding the orientations (i.e., compared to the $\kappa = 0$ case), with the optimal performance being achieved at an orientation concentration setting of $\kappa = 100$. Note that the values of $\kappa = \{0, 10, 20, 35, 50, 70, 100, 150, 200\}$ correspond to angular standard deviations of approximately $\{\infty, 18.1, 12.8, 9.7, 8.1, 6.8, 5.7, 4.7, 4.1\}$ degrees, respectively. As seen in Figure 7.8, using a higher out-of-plane variance for the positional ultrasound features also significantly improves registration accuracy compared to the isotropic case (i.e., compared to a standard deviation of 1 mm in the out-of-plane direction), with the optimal value being the standard deviation of 1.5 mm. The range of values for the out-of-plane variance in Figure 7.8 were tested using the optimal orientation concentration setting of $\kappa = 100$. Similarly, the range of values for the orientation concentration in Figure 7.7 where tested using the optimal out-of-plane variance setting of 1.5 mm standard deviation.

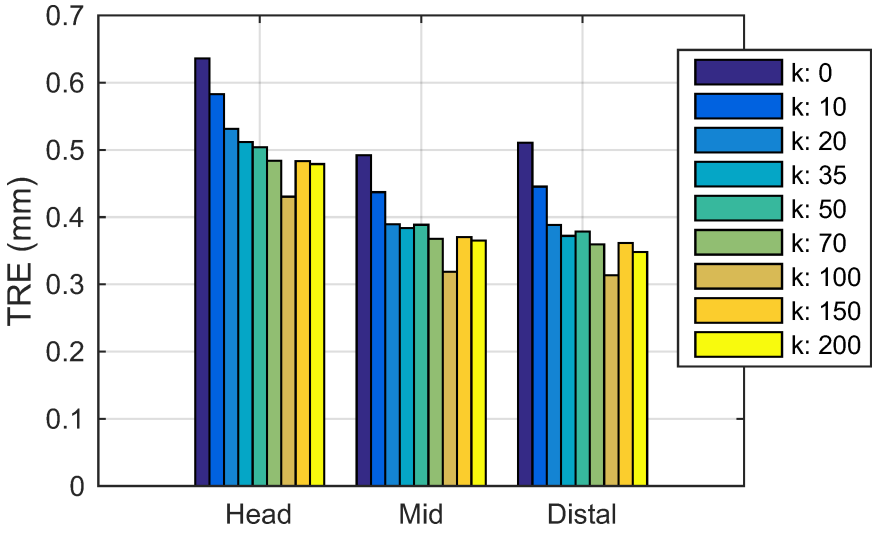

Figure 7.7: Variation in registration accuracy of the P-IMLOP algorithm with respect to the orientation concentration parameter, κ, for surface normal orientations segmented from ultrasound. For this analysis, the out-of-plane variance was set to its optimal setting of 1.5 mm standard deviation.

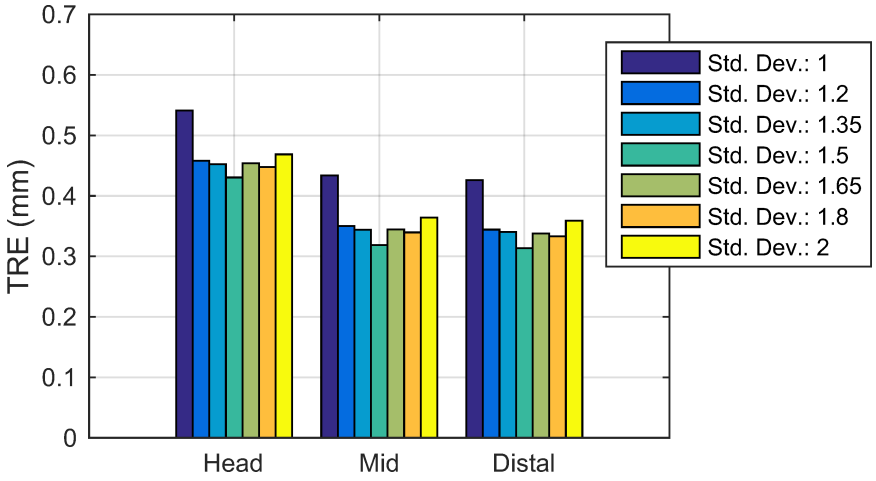

Figure 7.8: Variation in registration accuracy of the P-IMLOP algorithm with respect to the out-of-plane variance parameter for feature points segmented from ultrasound. For this analysis, the orientation concentration was set to its optimal setting of $\kappa = 100$.

As was noted earlier, the speed of sound was calibrated for these experiments, with the speed of sound through the tissue-mimicking plastisol medium having a calibrated value of 1373 m/s. As a final point of investigation, the sensitivity of the registration outcomes were assessed with respect to the accuracy of the assumed speed of sound setting, which may be imprecisely known during a real procedure involving a real patient. Figure 7.9 presents the outcome of this study, which evaluated the change in registration accuracy for speed of sound errors up to $\pm 2\%$ with respect to the calibrated value. As a point of comparison, the effect of changing the speed of sound was evaluated for both the P-IMLOP (Figure 7.9A) and the ICP (Figure 7.9B) algorithms using the same ultrasound data. As seen in the figures, the P-IMLOP algorithm achieves higher registration accuracy than ICP at all speed of sound settings. In this study, the registration errors for both algorithms approximately double for a speed of sound error of -2%, whereas the registration error ranges from approximately doubling to even improving for speed of sound errors up to $+2\%$, which varies by location on the bone.

7.7 Concluding Remarks

The P-IMLOP algorithm has been demonstrated as an effective method for feature-based registration of tracked ultrasound data, particularly in application to computer-assisted orthopedic surgery. Phantom-based experiments targeting computer-assisted

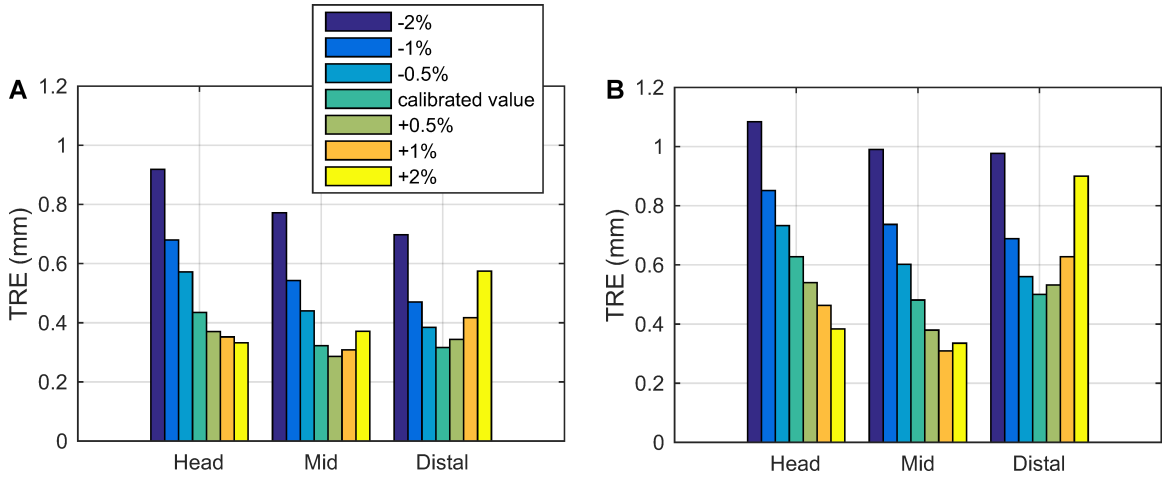

Figure 7.9: Change in registration accuracy relative to errors in the assumed speed of sound with respect to the calibrated speed of sound value for (A) the P-IMLOP and (B) the ICP algorithms.

THR surgery have demonstrated that P-IMLOP has potential to reduce the invasiveness and improve the registration accuracy of current computer-assisted THR procedures. A limitation of P-IMLOP in its current form is non-robustness to outliers, which is a topic for future work. Other topics for future work include online estimation of the speed-of-sound and more advanced noise-model settings, such as modeling tracker and segmentation uncertainties (such as accounting for additional segmentation error associated with varying angle-of-attack between the ultrasound beam and the bone surface) as well as modeling the change in the ultrasound imaging resolution at different imaging depths, particularly with respect to the elevational direction.

Another matter for future work is to assess the impact of normalizing the influence of the proximally sampled femur points compared to the influence of the ultrasound image features. Because the number of ultrasound image features far outnumber the 14 points sampled from the proximal femur, the influence of the proximally sampled

points may be quite small. To counter this effect, the covariances of the proximal points could be reduced in order to restore the influence of the proximal points. This method of normalization is investigated in the following chapter for a different clinical application and algorithm.

7.8 Contributions

5. Projected IMLOP (P-IMLOP) Algorithm[29]

   (a) a probabilistic algorithm for registering features from *tracked B-mode ultrasound* imaging; this algorithm registers *position* features characterized by *anisotropic* uncertainty and *projected orientation* features defined within arbitrarily oriented planes in 3D space

   (b) an efficient implementation of the P-IMLOP algorithm consisting of:

      i. *P-IMLOP Correspondence Phase*: a fast PD-tree-based search for efficiently computing the most likely matches from a model shape considering both the position and projected orientation components of the match error, as well as the anisotropic uncertainty in the position data

      ii. *P-IMLOP Registration Phase*: a gradient-based solution for the problem of registering projection-oriented data-shape features with a corresponding set of oriented model-shape features assuming anisotropic uncertainty in the feature positions; this solution is formed from the gradient equations of P-IMLOP's objective function, which is optimized using an off-the-shelf, nonlinear, BFGS quasi-Newton optimizer

   (c) design and construction of a human leg phantom for assessing registration accuracy in application to computer-assisted total hip replacement surgery

7.9 Published Work

Material from this chapter appeared in the following publication:

1. S. D. Billings, H. J. Kang, A. Cheng, E. M. Boctor, P. Kazanzides, and R. H. Taylor, "Minimally invasive registration for computer-assisted orthopedic surgery: Combining tracked ultrasound and bone surface points via the P-IMLOP algorithm," *International Journal of Computer Assisted Radiology and Surgery*, vol. 10, no. 6, pp. 761–771, 2015. [Online]. Available: `http://dx.doi.org/10.1007/s11548-015-1188-z`

Chapter 8

Video Iterative Most Likely Oriented Point (V-IMLOP) Algorithm

This chapter describes the Video Iterative Most Likely Oriented Point (V-IMLOP) algorithm, which is a special purpose algorithm for registering endoscopic video data. The V-IMLOP algorithm registers two types of video feature data: 3D point features known to scale, which are computed by *structure from motion* (SFM), and 2D oriented-point features representing *surface contours* in the video images. The scaled 3D positions of the SFM features are computed by tracking the positions of 2D image features through multiple video frames. Feature matching between video frames may be accomplished using the scale invariant feature transform (SIFT) developed

by Lowe[109] or a variant method (e.g., [110,111]). Given a set of 2D feature locations, which are tracked through multiple video frames, both the camera pose (position and orientation) associated with each video frame and the 3D positions of the tracked features may be estimated. This process of computing the scene geometry and camera poses is known as the structure from motion problem.[112–116] Since the feature positions are computed based on triangulation, these positions can be determined only to scale.[116] The V-IMLOP algorithm is independent of the method used to compute the feature positions, however, and any method other than that described above for computing 3D feature positions from video could be used. The 2D contour features represent the positions and normal orientations of occluding surface contours that appear in the video images. Unlike the SFM features, each contour feature is computed from only a single video image and is therefore a 2D rather than 3D feature. Obtaining these features requires segmenting the contours from the video images.

The component of the V-IMLOP algorithm that deals with registering the 2D contour features requires defining a geometric model of the imaging process in order to estimate the visible contours of the model shape for a given camera pose corresponding to each image, as well as in order to compute how the registration parameters affect the motion of the model-shape contours that project onto each imaging plane. An in-depth discussion concerning the geometric model for perspective cameras that is described here may be found in [116]. The V-IMLOP algorithm assumes that the imaging process that generated the video images follows a pinhole camera model.

The basic assumption of this model is that all light rays pass undeflected through a single point located at the center of the camera's aperture, which is the optical center position of the camera. A 2D illustration of the pinhole camera model is provided in Figure 8.1, with the 3D case being a trivial extension of this illustration. Figure 8.1 shows the perspective projection equation for projecting a point feature onto the imaging plane of the camera. This figure illustrates the true physical model of the camera, where the imaging plane is situated behind the lens at the focal length distance, f. Alternatively, a perspective projection may be modeled by situating a virtual imaging plane in front of the camera's optical center at the same distance, f, as shown in Figure 8.2. In this case, the perspective projection equations are simply negated compared to the physical camera model, as shown in Equations (8.1) and (8.2)

$$x_{\text{im}} = f\frac{X}{Z} \tag{8.1}$$

$$y_{\text{im}} = f\frac{Y}{Z} \tag{8.2}$$

where (X, Y, Z) is the 3D location of the feature in *camera coordinates* and (x_{im}, y_{im}) is the 2D location of the perspective-projected feature in metric *image coordinates*. As shown in Figure 8.2, the *camera coordinates* are defined as 3D coordinates whose origin is located at the optical center with the positive z-axis pointing along the depth direction, and the positive x-axis pointing to the right relative to the camera's imaging

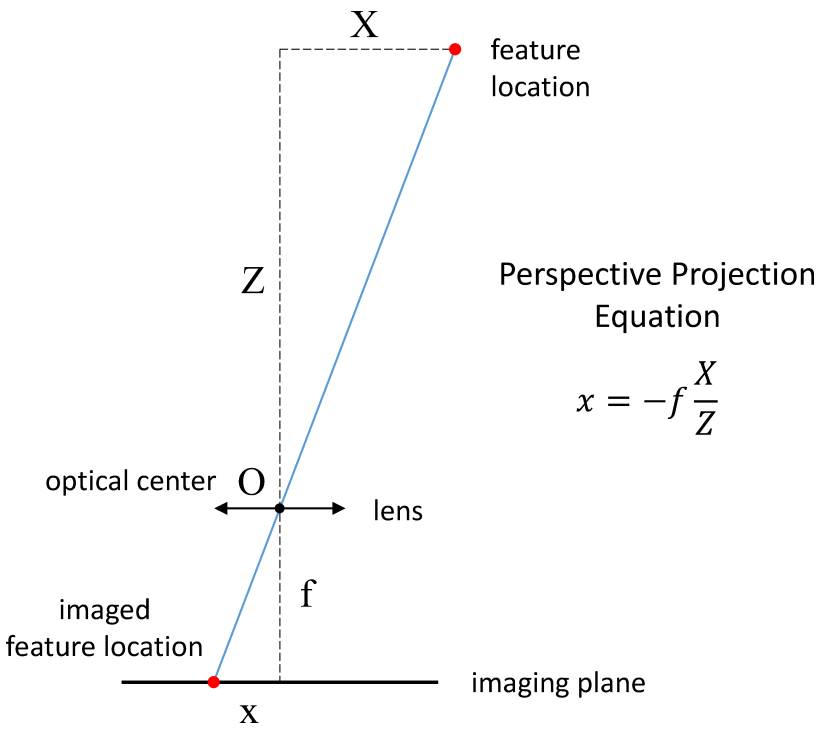

Figure 8.1: 2D illustration of the pinhole camera model, illustrating the perspective projection of a spatial feature onto the physical imaging plane of the camera, which is situated behind the camera lens. In this illustration, X and Z represent the spatial position of the feature in camera coordinates, where Z is the feature depth and where the origin is located at the optical center, O, while x denotes the projected position of the feature onto the imaging plane. The focal length of the camera is represented by f.

plane. The *image coordinates* are 2D coordinates defined on the imaging plane, with the positive x- and y-axis pointing right and down, respectively, within the image. Equations (8.1) and (8.2) are the perspective projection equations that are used by the V-IMLOP algorithm.

As alluded to above, in order to compute the 2D position where a 3D feature from the scene projects onto the imaging plane of the camera, the feature's location within the scene must first be described with respect to the camera's standard coordinate system (i.e., the 3D feature must be transformed to camera coordinates from what-

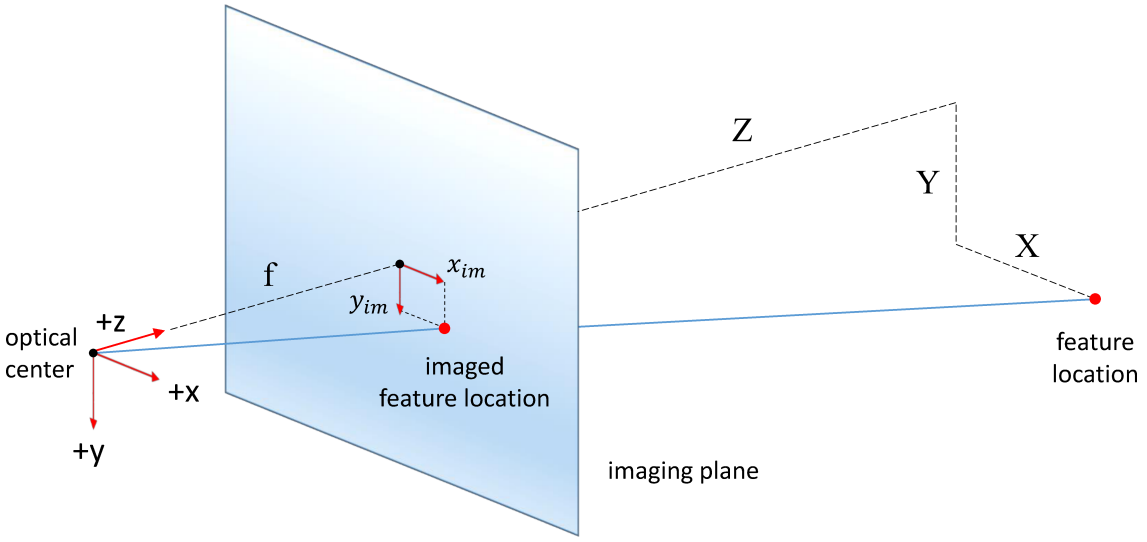

Figure 8.2: 3D illustration of the pinhole camera model, illustrating the perspective projection of a spatial feature onto a virtual imaging plane, which is situated in front of the camera. In this illustration, (X, Y, Z) defines the 3D position of the feature in camera coordinates; the camera coordinate frame has its origin located at the optical center, O, and its positive z-axis pointing along the depth direction. The 2D image coordinates of the perspective-projected feature are represented by $(x_{\text{im}}, y_{\text{im}})$, with the origin being located at the projected position of the optical center and the positive x- and y-axis pointing right and down, respectively, within the image plane. The focal length of the camera is denoted by f.

ever world coordinate system is being used to define the locations of objects within the scene). The rigid-body transformation, $\mathbf{T}_{\text{ext}} = [\mathbf{R}_{\text{ext}}, \mathbf{t}_{\text{ext}}]$, that accomplishes this transformation of a 3D point from world to camera coordinates is called the *extrinsic parameters* of the camera. The *intrinsic parameters* of the camera define the projection and transformation parameters that control how a 3D feature described in camera coordinates projects onto the 2D image plane. The perspective projection from camera coordinates to metric image coordinates is fully described by the focal length, f, which is one of the intrinsic parameters. However, locations on the image plane are generally described in units of pixels with the origin positioned in the upper left hand corner of the image. Conversion from metric to pixel units is controlled by the intrinsic scaling parameters, s_{x} and s_{y}, which define the number of pixels per metric unit (i.e., the inverse of the pixels' metric dimensions); these two scaling parameters allow for different pixel dimensions in the horizontal and vertical image directions, respectively (for the case of square pixels, $s_{\text{x}} = s_{\text{y}}$). In addition, a skew parameter, s_θ, may be introduced to allow for non-rectangular pixel geometries; however, the skew parameter is not required by most practical applications and is not considered here. Finally, relocating the pixel origin from the projected optical center position to the upper left corner of the image is controlled by the intrinsic translation parameters, o_{x} and o_{y}, which define the pixel offset between the projected optical center position and the pixel origin along the x- and y-axis of the image, respectively.

In computer vision applications, the equations for applying the extrinsic and in-

trinsic camera parameters are generally represented using the homogeneous notation for coordinate vectors and rigid-body transformations, which involves adding an extra dimension containing the value 1 to each coordinate vector and which combines the rigid-body transformation parameters for rotation and translation into a single 4×4 matrix. Equation (8.3) shows how the perspective projection equations of (8.1) and (8.2) are represented in homogeneous form.

$$Z \begin{bmatrix} x_{\text{im}} \\ y_{\text{im}} \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{8.3}$$

Equation (8.4) shows how the extrinsic and intrinsic camera parameters are sequentially applied using the homogeneous form in order to first transform a 3D feature from world coordinates to camera coordinates, then project the feature to the 2D image, and then finally to transform from metric image coordinates to pixel coordinates. Generally, the depth of the 3D point in camera coordinates (Z) is unknown; therefore, Equation (8.4) is commonly shown having an arbitrary depth value, represented by λ, in place of Z at the left hand side of the equation.

$$Z \begin{bmatrix} x_{\text{px}} \\ y_{\text{px}} \\ 1 \end{bmatrix} = \begin{bmatrix} s_{\text{x}} & 0 & o_{\text{x}} \\ 0 & s_{\text{y}} & o_{\text{y}} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{\text{ext}} & \mathbf{t}_{\text{ext}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_{\text{w}} \\ Y_{\text{w}} \\ Z_{\text{w}} \\ 1 \end{bmatrix} \tag{8.4}$$

Since the notation used in this dissertation favors functional operators over the homogeneous form, the functional forms of Equations (8.5) and (8.7) are used in place of Equations (8.3) and (8.4) for performing perspective projection and for applying the extrinsic and intrinsic camera parameters, respectively.

$$\begin{bmatrix} x_{\text{im}} \\ y_{\text{im}} \end{bmatrix} = f\,\Pi([\,X,\,Y,\,Z\,]^{\mathsf{T}})) \tag{8.5}$$

$$\Pi\left([\,x,\,y,\,z\,]^{\mathsf{T}}\right) = [\,x/z,\,y/z\,]^{\mathsf{T}} \tag{8.6}$$

$$\begin{bmatrix} x_{\text{px}} \\ y_{\text{px}} \end{bmatrix} = f\mathbf{S}\,\Pi(\mathrm{T}_{\text{ext}}([\,X_{\text{w}},\,Y_{\text{w}},\,Z_{\text{w}}\,]^{\mathsf{T}})) + \begin{bmatrix} o_{\text{x}} \\ o_{\text{y}} \end{bmatrix} \tag{8.7}$$

$$\mathbf{S} = \begin{bmatrix} s_{\text{x}} & 0 \\ 0 & s_{\text{y}} \end{bmatrix} \tag{8.8}$$

The operator, $\Pi(\cdot)$, of Equation (8.6) is referred to as the *perspective projection operator*. The scale factors for unit conversion are accommodated by the *unit conversion matrix*, $\mathbf{S}$, which is defined in Equation (8.8). For implementing the various software components of the V-IMLOP algorithm, it has been advantageous to take a middle ground between the image and pixel coordinates that are defined by Equations (8.5) and (8.7), respectively. This middle ground coordinate system uses units of pixels but keeps the origin located at the projected optical center position, as defined by

Equation (8.9).

$$\begin{bmatrix} x_{\text{im\_px}} \\ y_{\text{im\_px}} \end{bmatrix} = f\mathbf{S}\,\Pi(\mathrm{T}_{\text{ext}}([\,X_{\text{w}},\,Y_{\text{w}},\,Z_{\text{w}}\,]^{\mathsf{T}})) \tag{8.9}$$

In the remainder of this chapter, it is assumed that all 2D image features are described in the coordinate system defined by Equation (8.9).

8.1 Clinical Perspective

While several clinical applications exist for registering endoscopic video, the V-IMLOP algorithm is particularly motivated by the clinical application of endoscopic endonasal skull base surgery. An in-depth discussion of registration and its associated challenges within the context of this application is described by Mirota.[117] The endoscopic endonasal approach (EEA) for skull base surgery is a minimally invasive technique for performing surgery within the skull base region, such as for removal of tumors. In EEA surgery, internal anatomy is visualized by the surgeon using an endoscope paired with a camera and a digital monitor for visualization of the operative field. The targeted anatomy is accessed by inserting the endoscope and surgical tools through the nasal airway of the sinus, which has a complex anatomy shown in Figure 8.3. This minimally invasive approach offers several tangible benefits for the patient including faster recovery, lack of external incision, decreased trauma to normal tissue from surgery, and low patient morbidity.[118–120]

However, skull base surgeries remain technically challenging for surgeons. Skull

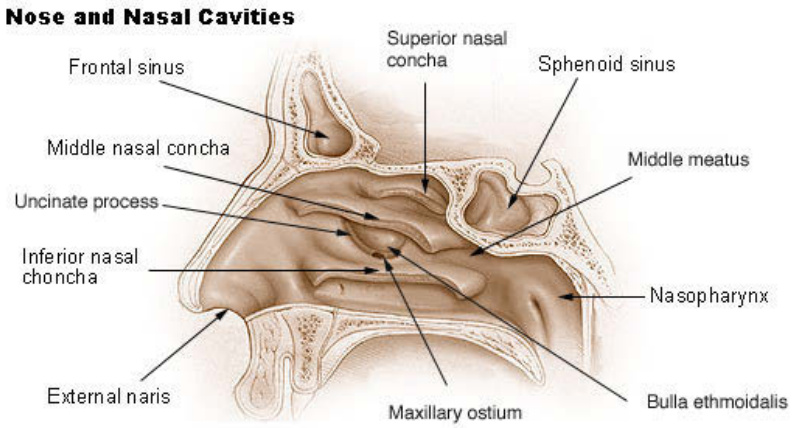

Figure 8.3: Anatomy of the sinus. "Illu nasal cavities". Licensed under Public Domain via Wikipedia - `https://en.wikipedia.org/wiki/File:Illu_nasal_cavities.jpg`

base surgeries are complex procedures characterized by complex anatomy and by critical anatomical structures within the skull base region, such as the carotid artery and cranial nerves. Because the surgical pathway and the targeted tissue lie in close proximity to such critical structures, exact knowledge of patient anatomy is required.[118, 120] Under the minimally invasive approach, the endoscopic view limits depth perception and the ability to visualize surrounding anatomy, which may increase navigational challenges. Navigation systems have therefore been proposed as an aid to the surgeon, which may further ensure the accuracy and safety of these procedures.[121–124] By combining a navigation system for tracking surgical instruments with a registration method for mapping the coordinates of the tracking system to the patient anatomy as defined in preoperative imaging, such as CT, the navigational awareness of the surgeon may be improved. A navigation system enables visualizing the positions of the surgical instruments relative to the anatomy shown

in preoperative imaging, thereby enabling the surgeon to more accurately implement the surgical approach and to continuously monitor the proximity of the surgical instruments relative to critical anatomical structures. Clearly, the effectiveness of a navigation system for accomplishing these aims is directly dependent on the ability of the system to accurately represent the locations of tracked instruments relative to the anatomical model of the patient. The accuracy of surgical guidance is impacted by both the accuracy of the navigational tracking system and by the accuracy of the registration used to define a spatial mapping between the coordinates of the tracking device and the anatomical coordinates defined by the preoperative imaging.

Direct image-based registration of the endoscopic video data is one method that has been proposed to improve the accuracy of surgical guidance for endoscopic applications, thereby eliminating the need for external tracking and its associated inaccuracies.[125] Alternatively, image-based registration could be applied in concert with external tracking in order to enhance the robustness and accuracy of both methods. A further advantage of a direct image-based navigation approach is that intraoperative shift in the true registration solution, such as due to patient motion, is automatically compensated. Image-based registration methods for EEA skull base surgery have been investigated by Burschka et al.[126] and by Mirota et al.[24,127] with promising results. Burschka et al.[126] demonstrated a SLAM-based technique for registering the endoscopic images using SSD tracking. Mirota et al.[24] incorporated a SIFT-based method for tracking features in sequential video images in order to reconstruct the

scaled 3D positions of these features using SFM [112]. The scaled 3D points are then registered to a preoperative model of the sinus surface using a variant of Trimmed ICP[128] that accounts for estimation of scale.[129]

The aim of developing the V-IMLOP algorithm is to build upon the prior work of Mirota et al.[24] in order to improve the robustness and accuracy of the computed registration by using an enhanced registration algorithm to register an extended class of video features. Like the method of Mirota et al., V-IMLOP incorporates 3D feature positions known to scale, which are computed using SFM. Unlike the registration method of Mirota et al., V-IMLOP also incorporates oriented 2D features representing the surface contours from the video images and supports an anisotropic framework for modeling anisotropic uncertainties in both types of feature data. The SFM and contour video features are registered to a preoperative CT image of the patient by segmenting the sinus airway from the CT image and converting the segmentation into a surface mesh modeling the interior wall of the sinus. This surface mesh of the interior sinus wall forms the model shape for registering the SFM and contour features, which together comprise the data shape. As described earlier, SFM and edge segmentation techniques are required in order to compute these features from the stream of video images. The contributions of this work are not in this area, however; thus, techniques for computing these features will not be further addressed. The work described in this chapter focuses on the development of the V-IMLOP registration algorithm for registering the video features after they have been obtained.

8.2 Probabilistic Model

The V-IMLOP algorithm incorporates a probabilistic framework formulated using 3D anisotropic Gaussian distributions to model the uncertainty of the 3D SFM feature point locations; the uncertainty of the 2D contour features are modeled using 2D anisotropic Gaussian and 2D von Mises[85] distributions for the contour positions and orientations, respectively. The V-IMLOP algorithm computes a similarity transformation ($\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$) for the registration, which extends a rigid-body transformation by adding an unknown scale factor as defined in Appendix A. Including the scale parameter is necessary because the reconstructed 3D video features are known only to scale.

The component of the probabilistic model that deals with the 3D position features that are computed by SFM is now discussed. Assuming zero-mean uncertainty and independence between these features, then the *match likelihood function* for an SFM data point, $\mathbf{x}_{\text{sfm}}$, given a current registration estimate, $\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$, is defined as

$$f_{\text{match}}(\mathbf{x}_{\text{sfm}} \,|\, \mathbf{y}_{\text{sfm}}, \boldsymbol{\Sigma}_{\text{sfm}}, s, \mathbf{R}, \mathbf{t}) = \frac{1}{(2\pi)^{3/2}|\boldsymbol{\Sigma}_{\text{sfm}}|^{1/2}} \cdot e^{-\frac{1}{2}(\mathbf{y}_{\text{sfm}} - s\mathbf{R}\mathbf{x}_{\text{sfm}} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_{\text{sfm}}^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{\text{sfm}} - s\mathbf{R}\mathbf{x}_{\text{sfm}} - \mathbf{t})} \quad (8.10)$$

where $\mathbf{y}_{\text{sfm}}$ (treated as the *mean position*) is a 3D point on the model shape that is assumed to be in correspondence with the transformed SFM data point, $\mathrm{T}(\mathbf{x}_{\text{sfm}}) = s\mathbf{R}\mathbf{x}_{\text{sfm}} + \mathbf{t}$, and where $\boldsymbol{\Sigma}_{\text{sfm}}$ is the *covariance matrix* of positional uncertainty for the

non-transformed SFM data point; for the transformed SFM data point, the covariance becomes $\mathbf{R}\boldsymbol{\Sigma}_{\text{sfm}}\mathbf{R}^{\mathsf{T}}$.

The component of the probabilistic model that deals with the 2D video contour features is now discussed. Assuming zero-mean uncertainty and independence between these features and between the position and orientation components of each feature, then the *match likelihood function* for a contour feature, $\mathbf{x}_{\text{2d}} = (\mathbf{x}_{\text{2dp}}, \hat{\mathbf{x}}_{\text{2dn}})$, given a current registration estimate, $\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$, is defined as

$$f_{\text{match}}(\mathbf{x}_{\text{2d}} \,|\, \mathbf{y}_{\text{3d}}, \boldsymbol{\Sigma}_{\text{2d}}, \kappa, s, \mathbf{R}, \mathbf{t}) = \frac{1}{(2\pi)^2 |\boldsymbol{\Sigma}_{\text{2d}}|^{1/2}\, \text{I}_0(\kappa)}$$
$$\cdot\, e^{\kappa \frac{\mathbf{S}\,\text{P}_{\text{xy}}(\mathbf{R}_{\text{cam}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{\text{3dn}})^{\mathsf{T}}}{\|\mathbf{S}\,\text{P}_{\text{xy}}(\mathbf{R}_{\text{cam}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{\text{3dn}})\|}\hat{\mathbf{x}}_{\text{2dn}} - \frac{1}{2}(f\mathbf{S}\,\Pi(\text{T}_{\text{cam}}(\frac{1}{s}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{\text{3dp}}-\mathbf{t}))) - \mathbf{x}_{\text{2dp}})^{\mathsf{T}}\boldsymbol{\Sigma}_{\text{2d}}^{-1}(f\mathbf{S}\,\Pi(\text{T}_{\text{cam}}(\frac{1}{s}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{\text{3dp}}-\mathbf{t}))) - \mathbf{x}_{\text{2dp}})} \tag{8.11}$$

where $\boldsymbol{\Sigma}_{\text{2d}}$ is the *covariance matrix* of uncertainty in the 2D position of the contour feature, $\mathbf{x}_{\text{2dp}}$, and where κ is the *concentration parameter* of uncertainty in the 2D orientation of the contour feature, $\hat{\mathbf{x}}_{\text{2dn}}$. $\mathbf{y}_{\text{3d}} = (\mathbf{y}_{\text{3dp}}, \hat{\mathbf{y}}_{\text{3dn}})$ is a 3D oriented point on the model shape that is assumed to be in correspondence with the 2D perspective projected contour feature, $\mathbf{x}_{\text{2d}} = (\mathbf{x}_{\text{2dp}}, \hat{\mathbf{x}}_{\text{2dn}})$. In the likelihood function of Equation (8.11), the registration transformation is applied to the model-shape feature, $\mathbf{y}_{\text{3d}}$, rather than to the contour feature (i.e., the data-shape feature), $\mathbf{x}_{\text{2d}}$, because the contour features are segmented from 2D video images and therefore represent 2D perspective projections of the 3D data. The model features, on the other hand,

exist in the proper 3D form that is required for being transformed by the registration. Since the registration maps points from data to model coordinates, the inverse of the registration is needed when applying the registration to features of the model. The expressions for the inverse registration applied to the model feature, $\mathbf{y}_{3\mathrm{d}}$, appear in Equation (8.11) and are defined as

$$\mathrm{T}^{-1}(\mathbf{y}_{3\mathrm{dp}}) = \frac{1}{s}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{3\mathrm{dp}} - \mathbf{t}) \tag{8.12}$$

$$\mathrm{T}^{-1}(\hat{\mathbf{y}}_{3\mathrm{dn}}) = \mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\mathrm{dn}} \tag{8.13}$$

where the transformation of the position and orientation components of, $\mathbf{y}_{3\mathrm{d}}$, are shown separately.

The position term of Equation (8.11) is now discussed, which involves the position component, $\mathbf{x}_{2\mathrm{dp}}$, of the data feature, and the position component, $\mathbf{y}_{3\mathrm{dp}}$, of the model feature. The 3D model feature position, $\mathbf{y}_{3\mathrm{dp}}$, is passed through a series of operations before being subtracted from the 2D data feature position, $\mathbf{x}_{2\mathrm{dp}}$. The first operation transforms the model point by the inverse of the current registration estimate, $\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$, which positions the model point in 3D data coordinates. The model point is then transformed to the local 3D coordinates of the camera by the rigid-body transformation, $\mathbf{T}_{\mathrm{cam}}$. $\mathbf{T}_{\mathrm{cam}}$ represents the extrinsic camera parameters of the video image from which the contour feature, $\mathbf{x}_{2\mathrm{d}} = (\mathbf{x}_{2\mathrm{dp}}, \hat{\mathbf{x}}_{2\mathrm{dn}})$, was taken. Following transformation of the model point to the local coordinate frame of the camera, the

perspective projection operator, Π, is applied to the model point, which projects the model point to the 2D plane of the video image in order to align the model feature with the 2D video contour. This perspective projection operator assumes that the local camera coordinates are defined such that the camera view direction points along the positive z-axis, as indicated by its definition in Equation (8.6) (and as illustrated in Figure 8.2). Finally, the perspective projection of the model point is completed by multiplying by the camera focal length, f, and the projected location is converted from metric to pixel units by the diagonal unit conversion matrix, $\mathbf{S}$, which was defined earlier in Equation (8.8).

The orientation term of Equation (8.11) is now discussed, which involves the orientation component, $\hat{\mathbf{x}}_{2\text{dn}}$, of the data feature and the orientation component, $\hat{\mathbf{y}}_{3\text{dn}}$, of the model feature. In similar manner as described for the position term, the model orientation is first transformed by the registration inverse, in order to obtain the model orientation in 3D data coordinates, and then transformed by the camera extrinsic parameters, in order to obtain the model orientation in local camera coordinates. An important difference compared to the positional term is that an orthographic projection, P_{xy}, is used to project the model orientation to the video image plane rather than the perspective projection, Π.

$$\mathrm{P}_{\text{xy}}\left([\,x,\,y,\,z\,]^{\mathsf{T}}\right) = [\,x,\,y\,]^{\mathsf{T}} \tag{8.14}$$

This orthographic projection is used because when the 3D model orientation is located parallel to the image plane, then the perspective projection operator blows up due to division by a depth value of zero, whereas the orthographic projection is stable in this case. For orientations that are sufficiently parallel to the image plane, the two projection operators otherwise give equivalent results, because the model orientation is renormalized to restore unit length following the projection to the image plane and following the conversion to pixel units by $\mathbf{S}$.

In the correspondence phase of the V-IMLOP algorithm, a match for each data feature is computed from the model by maximizing the match likelihood function of Equation (8.10) if the data feature is an SFM feature or of Equation (8.11) if the data feature is a contour feature.

For an SFM data feature, maximizing the match likelihood function of Equation (8.10) reduces to computing the point, $\mathbf{y}_{\text{sfm}}$, from anywhere on the model shape that minimizes the following *SFM match error function*

$$\mathrm{E}_{\text{V-IMLOP,sfm}}(\mathbf{x}_{\text{sfm}}, \mathbf{y}_{\text{sfm}}, \boldsymbol{\Sigma}_{\text{sfm}}, s, \mathbf{R}, \mathbf{t}) = \frac{1}{2}(\mathbf{y}_{\text{sfm}} - s\mathbf{R}\mathbf{x}_{\text{sfm}} - \mathbf{t})^{\mathsf{T}}\mathbf{R}\boldsymbol{\Sigma}_{\text{sfm}}^{-1}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{\text{sfm}} - s\mathbf{R}\mathbf{x}_{\text{sfm}} - \mathbf{t}) \ . \quad (8.15)$$

For a contour data feature, maximizing the match likelihood function of Equation (8.11) with respect to the oriented model point, $\mathbf{y}_{\text{3d}} = (\mathbf{y}_{\text{3dp}}, \hat{\mathbf{y}}_{\text{3dn}})$, reduces to

minimizing the following *contour match error function*

$$\mathrm{E}_{\text{V-IMLOP,ctr}}(\mathbf{x}_{2\mathrm{d}}, \mathbf{y}_{3\mathrm{d}}, \boldsymbol{\Sigma}_{2\mathrm{d}}, \kappa, s, \mathbf{R}, \mathbf{t}) =$$

$$\frac{1}{2}(f\mathbf{S}\,\Pi(\mathrm{T}_{\mathrm{cam}}(\frac{1}{s}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{3\mathrm{dp}} - \mathbf{t}))) - \mathbf{x}_{2\mathrm{dp}})^{\mathsf{T}}\boldsymbol{\Sigma}_{2\mathrm{d}}^{-1}(f\mathbf{S}\,\Pi(\mathrm{T}_{\mathrm{cam}}(\frac{1}{s}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{3\mathrm{dp}} - \mathbf{t}))) - \mathbf{x}_{2\mathrm{dp}})$$
$$- \kappa\frac{\mathbf{S}\,\mathrm{P}_{\mathrm{xy}}(\mathbf{R}_{\mathrm{cam}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\mathrm{dn}})^{\mathsf{T}}}{\|\mathbf{S}\,\mathrm{P}_{\mathrm{xy}}(\mathbf{R}_{\mathrm{cam}}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\mathrm{dn}})\|}\hat{\mathbf{x}}_{2\mathrm{dn}} \quad . \quad (8.16)$$

However, the oriented model point, $\mathbf{y}_{3\mathrm{d}} = (\mathbf{y}_{3\mathrm{dp}}, \hat{\mathbf{y}}_{3\mathrm{dn}})$, must be a point that projects to a visible model-shape contour in the video image. Therefore, the matches cannot simply be chosen as any point on the model shape that maximizes Equation (8.11), but rather must be chosen from the set of model-shape points that form visible occluding contours when projected to the video image plane. The correspondence phase for these features is therefore implemented as a two-stage procedure. In the first stage, the points on the model shape that form visible occluding contours within each video image are computed. In order to compute the projected model-shape contours, the current estimate of the registration is used to estimate the extrinsic parameters of the camera associated with each video image. The set of projected model-shape contours that are computed for the jth video image are denoted by Ψ_j. In the second stage, the contour features within each jth video image are matched only to the projected model-shape contours, Ψ_j, associated with that video image. Since the *projected* model-shape contours are used for matching, the matching is performed within the 2D plane of the video image rather than in 3D coordinates. Therefore, minimizing

the contour match error equation of Equation (8.11), subject to the visible contour constraint for the jth video image, reduces to computing the projected model-shape point, $(\mathbf{y}_{2\mathrm{dp}}, \hat{\mathbf{y}}_{2\mathrm{dn}})$, from the set Ψ_j that minimizes the following *projected contour match error function*

$$\mathrm{E}_{\text{V-IMLOP,2d}}(\mathbf{x}_{2\mathrm{d}}, \mathbf{y}_{2\mathrm{d}}, \boldsymbol{\Sigma}_{2\mathrm{d}}, \kappa) = \frac{1}{2}(\mathbf{y}_{2\mathrm{dp}} - \mathbf{x}_{2\mathrm{dp}})^{\mathsf{T}} \boldsymbol{\Sigma}_{2\mathrm{d}}^{-1} (\mathbf{y}_{2\mathrm{dp}} - \mathbf{x}_{2\mathrm{dp}}) - \kappa \hat{\mathbf{y}}_{2\mathrm{dn}}^{\mathsf{T}} \hat{\mathbf{x}}_{2\mathrm{dn}} \ . \quad (8.17)$$

After computing the projected model-shape contour feature, $\mathbf{y}_{2\mathrm{d}} = (\mathbf{y}_{2\mathrm{dp}}, \hat{\mathbf{y}}_{2\mathrm{dn}})$, that is in correspondence with the data-shape contour feature, $\mathbf{x}_{2\mathrm{d}} = (\mathbf{x}_{2\mathrm{dp}}, \hat{\mathbf{x}}_{2\mathrm{dn}})$, it is necessary to also compute the 3D location on the model that produced the projected model-shape feature. This 3D model-shape feature, denoted as $\mathbf{y}_{3\mathrm{d}} = (\mathbf{y}_{3\mathrm{dp}}, \hat{\mathbf{y}}_{3\mathrm{dn}})$, will be needed by the registration phase in order to properly compute the shift in the projected feature with respect to changes in the registration. Note that the transformation parameters, $\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$, do not appear in Equation (8.17), because the transformation has already been incorporated into the projected model-shape feature, $\mathbf{y}_{2\mathrm{d}} = (\mathbf{y}_{2\mathrm{dp}}, \hat{\mathbf{y}}_{2\mathrm{dn}})$, which has the following definition

$$\mathbf{y}_{2\mathrm{dp}} = f\mathbf{S}\,\Pi(\mathrm{T}_{\mathrm{cam}j}(\frac{1}{s}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{3\mathrm{dp}} - \mathbf{t}))) \quad (8.18)$$

$$\hat{\mathbf{y}}_{2\mathrm{dn}} = \frac{\mathbf{S}\,\mathrm{P}_{\mathrm{xy}}(\mathbf{R}_{\mathrm{cam}j}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\mathrm{dn}})}{\|\mathbf{S}\,\mathrm{P}_{\mathrm{xy}}(\mathbf{R}_{\mathrm{cam}j}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{3\mathrm{dn}})\|} \ . \quad (8.19)$$

In the registration phase of the V-IMLOP algorithm, an updated pose for the data points is determined by computing the similarity transformation, $\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$, that maximizes the total match likelihood over all the data features and model correspondences, which simplifies to minimizing the *total match error* shown below

$$\begin{aligned}\mathbf{T} = \underset{[s,\mathbf{R},\mathbf{t}]}{\operatorname{argmin}} \Bigg(& \frac{1}{2} \sum_{i=1}^{n_{\text{sfm}}} \mathbf{d}_{\text{sfm}i}^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_{\text{sfm}i}^{-1} \mathbf{R}^{\mathsf{T}} \mathbf{d}_{\text{sfm}i} + \frac{1}{2} \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} \mathbf{d}_{\text{ctr}ji}^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{2d}ji}^{-1} \mathbf{d}_{\text{ctr}ji} \\ & - \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} \kappa_{ji} \frac{\mathbf{S}\, \mathrm{P}_{\text{xy}}(\mathbf{R}_{\text{cam}j} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\text{3dn}ji})^{\mathsf{T}}}{\| \mathbf{S}\, \mathrm{P}_{\text{xy}}(\mathbf{R}_{\text{cam}j} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\text{3dn}ji}) \|} \hat{\mathbf{x}}_{\text{2dn}ji} \Bigg)\end{aligned} \tag{8.20}$$

$$\mathbf{d}_{\text{sfm}i} = \mathbf{y}_{\text{sfm}i} - s\mathbf{R}\mathbf{x}_{\text{sfm}i} - \mathbf{t}$$

$$\mathbf{d}_{\text{ctr}ji} = f\mathbf{S}\, \Pi(\mathrm{T}_{\text{cam}j}(\frac{1}{s}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{\text{3dp}ji} - \mathbf{t}))) - \mathbf{x}_{\text{2dp}ji}$$

where n_{sfm} is the number of SFM features, n_{cam} is the number of video images, and $n_{\text{ctr}j}$ is the number of contour features within the jth video image. The model-shape features $\mathbf{y}_{\text{3d}ji} = (\mathbf{y}_{\text{3dp}ji}, \hat{\mathbf{y}}_{\text{3dn}ji})$ are the 3D model-shape features that are affiliated with the 2D projected model-shape features, $\mathbf{y}_{\text{2d}ji} = (\mathbf{y}_{\text{2dp}ji}, \hat{\mathbf{y}}_{\text{2dn}ji})$, that were chosen as the matches in the preceding correspondence phase.

8.3 Algorithm Overview

In this section, a high-level overview of the V-IMLOP algorithm is presented, followed by later sections that detail the two key sub-phases of V-IMLOP: the *correspondence phase* for computing matches on the model shape that correspond to the

data-shape features for a given registration estimate and the *registration phase* for computing the optimal alignment between these matched features in order to update the registration estimate. A summary of the V-IMLOP algorithm is provided as Algorithm 8.1. In addition to the computations for the correspondence and registration phases that were described in the earlier section, the V-IMLOP algorithm also includes computations for outlier rejection and for enforcing anatomical constraints on the estimated camera positions and scale.

The computations that were described earlier in Section 8.2 concerning the correspondence and registration phases occur in Steps 3–6 and Step 17, respectively, of Algorithm 8.1. Note that matching of the video contours in Step 5 includes support for an optional constraint on the maximum orientation match error, which can be useful to ensure that the video contours do not match to infeasible model-shape contours that have widely differing orientations. The threshold value, $\theta_{\text{match\_max}}$, that defines the maximum permitted orientation match error is an input to the algorithm that may be set by the user. While not explicitly stated in the algorithm summary, if no match is found in Step 5 that satisfies the orientation match error constraint for a given video contour feature, then that feature is flagged as invalid for the remainder of the current iteration and is not considered within the subsequent registration phase of Step 17.

The computations for outlier rejection occur in Steps 8–15 of the algorithm summary and are performed immediately following the correspondence phase. Feature

Algorithm 8.1. Video Iterative Most Likely Oriented Point (V-IMLOP)

input : SFM data features (3D points): $\mathbf{X}_{\text{sfm}} = \{\mathbf{x}_{\text{sfm}i}\}$
Contour data features (2D oriented points): $\mathbf{X}_{\text{ctr}} = \{(\mathbf{x}_{\text{2dp}ji}, \hat{\mathbf{x}}_{\text{2dn}ji})\}$
SFM data uncertainty parameters: $\{\mathbf{\Sigma}_{\text{sfm}i}\}$
Contour data uncertainty parameters: $\{\mathbf{\Sigma}_{\text{2d}ji}\}$ and $\{\kappa_{ji}\}$
Extrinsic camera parameters defined in data coordinates: $\{\mathbf{T}_{\text{cam}j}\}$
Intrinsic camera parameters: f and $\mathbf{S}$
Outlier trim ratio for SFM features: r_{trim} (default: 0%)
Outlier p-values for chi-square tests: p_{sfm}, p_{ctrp}, p_{ctrn} (defaults: 0.99)
Max outlier ratio for contour features: $r_{\text{outlier\_max}}$ (default: 1)
Max match angle for contour features: $\theta_{\text{match\_max}}$ (default: 180 deg.)
Disable outlier rejection for first iteration: $\varphi_{\text{disable\_1stIter}}$ (default: true)
Registration backup factor: $backupFactor$ (default: 0.75)
Model shape: Ψ
Initial transformation: $\mathbf{T}_0 = [s_0, \mathbf{R}_0, \mathbf{t}_0]$
Constraints on scale estimation: $[s_{\min}, s_{\max}]$ (default: $[-\infty, \infty]$)

output: Final transformation, $\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$, that aligns $\mathbf{X}_{\text{sfm}}$ and $\mathbf{X}_{\text{ctr}}$ with Ψ

Initialize: $\mathbf{T} \leftarrow \mathbf{T}_0\,, \quad \{\varphi_{\text{sfm}i}\} \leftarrow 1\,, \quad \{\varphi_{\text{ctr}ji}\} \leftarrow 1$

while *not converged* **do**

Compute the SFM feature matches from the model shape (Section 8.4.1):

$$\mathbf{y}_{\text{sfm}i} \leftarrow \underset{\mathbf{y}_{\text{sfm}} \in \Psi}{\text{argmin}} \left(\frac{1}{2} (\mathbf{y}_{\text{sfm}} - s\mathbf{R}\mathbf{x}_{\text{sfm}i} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \mathbf{\Sigma}_{\text{sfm}i}^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{\text{sfm}} - s\mathbf{R}\mathbf{x}_{\text{sfm}i} - \mathbf{t}) \right)$$

for $i = 1..n_{\text{sfm}}$

For each jth image, compute the set of projected model-shape points (Ψ_j) that form visible occluding contours within the image (Section 8.4.2.1)

For each jth image, compute matches for the video contour features; compute the matches from the set of model-shape contours in Ψ_j (Section 8.4.2.2):

$$(\mathbf{y}_{\text{2dp}ji}, \hat{\mathbf{y}}_{\text{2dn}ji}) \leftarrow \underset{(\mathbf{y}_{\text{2dp}}, \hat{\mathbf{y}}_{\text{2dn}}) \in \Psi_j}{\text{argmin}} \left(\frac{1}{2} (\mathbf{y}_{\text{2dp}} - \mathbf{x}_{\text{2dp}ji})^{\mathsf{T}} \mathbf{\Sigma}_{\text{2d}ji}^{-1} (\mathbf{y}_{\text{2dp}} - \mathbf{x}_{\text{2dp}ji}) - \kappa_{ji} \hat{\mathbf{y}}_{\text{2dn}}^{\mathsf{T}} \hat{\mathbf{x}}_{\text{2dn}ji} \right) \quad \text{for } j = 1..n_{\text{cam}},\ i = 1..n_{\text{ctr}j}$$

$$\text{subject to:} \quad \text{acos}(\hat{\mathbf{y}}_{\text{2dn}}^{\mathsf{T}} \hat{\mathbf{x}}_{\text{2dn}ji}) \leq \theta_{\text{match\_max}}$$

Store the pre-projection 3D locations of the contour matches

$$(\mathbf{y}_{\text{3dp}ji}, \hat{\mathbf{y}}_{\text{3dn}ji}) \leftarrow \text{ 3D location of the projected feature } (\mathbf{y}_{\text{2dp}ji}, \hat{\mathbf{y}}_{\text{2dn}ji})$$

for $j = 1..n_{\text{cam}},\ i = 1..n_{\text{ctr}j}$

Algorithm 8.1. V-IMLOP (Continued)

7 **if** $\varphi_{disable\_1stIter} ==$ true *and this is first iteration* **then** Goto Step 16

8 Trim the worst $r_{\text{trim}}\%$ of the SFM feature matches:

$$\varphi_{\text{sfm\_trim}i} \leftarrow \begin{cases} 0 & E_{\text{sfm}i} \in \text{upper } r_{\text{trim}}\% \text{ of SFM match errors} \\ 1 & \text{otherwise} \end{cases}$$

$$\text{where} \quad E_{\text{sfm}i} = \frac{1}{2}\mathbf{d}_{\text{sfm}i}^{\mathsf{T}}\mathbf{R}\boldsymbol{\Sigma}_{\text{sfm}i}^{-1}\mathbf{R}^{\mathsf{T}}\mathbf{d}_{\text{sfm}i}\,, \quad \mathbf{d}_{\text{sfm}i} = \mathbf{y}_{\text{sfm}i} - s\mathbf{R}\mathbf{x}_{\text{sfm}i} - \mathbf{t}$$

9 Compute the mean square match distance of the inlier SFM features:

$$\sigma_{\text{match\_sfm}}^2 \leftarrow \frac{1}{n_{\text{inliers\_sfm}}}\sum_{i=1}^{n_{\text{sfm}}}\varphi_{\text{sfm\_trim}i}\varphi_{\text{sfm}i}\|\mathbf{d}_{\text{sfm}i}\|^2$$

$$\text{where} \quad n_{\text{inliers\_sfm}} = \sum_{i=1}^{n_{\text{sfm}}}\varphi_{\text{sfm\_trim}i}\varphi_{\text{sfm}i}$$

10 Perform chi-square test on the SFM feature match errors:

$$\varphi_{\text{sfm\_chisqr}i} \leftarrow \begin{cases} 0 & \mathbf{d}_{\text{sfm}i}^{\mathsf{T}}\mathbf{R}(\boldsymbol{\Sigma}_{\text{sfm}i} + \sigma_{\text{match\_sfm}}^2\mathbf{I})^{-1}\mathbf{R}^{\mathsf{T}}\mathbf{d}_{\text{sfm}i} > \text{chi2inv}(p_{\text{sfm}}, 3) \\ 1 & \text{otherwise} \end{cases}$$

11 Compile the SFM feature outliers:

$$\varphi_{\text{sfm}i} \leftarrow \varphi_{\text{sfm\_trim}i}\varphi_{\text{sfm\_chisqr}i} \quad \text{for } i = 1..n_{\text{sfm}}$$

12 For each jth image, compute the mean square position and orientation match errors of the inlier contour features:

$$\sigma_{\text{match\_ctrp}j}^2 \leftarrow \frac{1}{n_{\text{inliers\_ctr}j}}\sum_{i=1}^{n_{\text{ctr}j}}\varphi_{\text{ctr}ji}\|\mathbf{d}_{\text{2dp}ji}\|^2$$

$$\sigma_{\text{match\_ctrn}j}^2 \leftarrow \frac{1}{n_{\text{inliers\_ctr}j}}\sum_{i=1}^{n_{\text{ctr}j}}\varphi_{\text{ctr}ji}d_{\text{2dn}ji}^2$$

$$\text{where} \quad n_{\text{inliers\_ctr}j} = \sum_{i=1}^{n_{\text{ctr}j}}\varphi_{\text{ctr}ji}$$

$$\mathbf{d}_{\text{2dp}ji} = \mathbf{y}_{\text{2dp}ji} - \mathbf{x}_{\text{2dp}ji}\,, \quad d_{\text{2dn}ji} = \text{acos}(\hat{\mathbf{y}}_{\text{2dn}ji}^{\mathsf{T}}\hat{\mathbf{x}}_{\text{2dn}ji})$$

13 Perform chi-square test on the contour feature position and orientation match errors:

$$\varphi_{\text{ctrp}ji} \leftarrow \begin{cases} 0 & \mathbf{d}_{\text{2dp}ji}^{\mathsf{T}}(\boldsymbol{\Sigma}_{\text{2d}ji} + \sigma_{\text{match\_ctrp}j}^2\mathbf{I}))^{-1}\mathbf{d}_{\text{2dp}ji} > \text{chi2inv}(p_{\text{ctrp}}, 2) \\ 1 & \text{otherwise} \end{cases}$$

$$\varphi_{\text{ctrn}ji} \leftarrow \begin{cases} 0 & d_{\text{2dn}ji}(\frac{1}{\kappa_{ji}} + \sigma_{\text{match\_ctrn}j}^2)^{-1}d_{\text{2dn}ji} > \text{chi2inv}(p_{\text{ctrn}}, 1) \\ 1 & \text{otherwise} \end{cases}$$

14 Compile the contour feature outliers:

$$\varphi_{\text{ctr}ji} \leftarrow \varphi_{\text{ctrp}ji}\varphi_{\text{ctrn}ji} \quad \text{for } j = 1..n_{\text{cam}}\,,\ i = 1..n_{\text{ctr}j}$$

Algorithm 8.1. V-IMLOP (Continued)

15 Restrict the percentage of outlying contour features within each image

foreach jth image **do**

if $\frac{\sum_{i=1}^{n_{ctrj}} \varphi_{ctrji}}{n_{ctrj}} > r_{outlier\_max}$ **then**

Sort the outliers in descending order by the sum of their position and orientation error values that were computed within the chi-square test in Step 12; for outliers occupying a sorted position beyond $r_{\text{outlier\_max}}$ percent of the total number of data features within the jth image, set $\varphi_{\text{ctr}ji} = 1$

end

end

16 $\mathbf{T}_{\text{prev}} \leftarrow \mathbf{T}$

17 Register all feature correspondences (Section 8.5):

$$\mathbf{T} = \underset{[s,\mathbf{R},\mathbf{t}]}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n_{\text{sfm}}} \varphi_{\text{sfm}i} \mathbf{d}_{\text{sfm}i}^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_{\text{sfm}i}^{-1} \mathbf{R}^{\mathsf{T}} \mathbf{d}_{\text{sfm}i} \right.$$

$$\left. + \frac{1}{2} \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} \varphi_{\text{ctr}ji} \mathbf{d}_{\text{ctr}ji}^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{2d}ji}^{-1} \mathbf{d}_{\text{ctr}ji} - \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} \kappa_{ji} \frac{\mathbf{S}\, \mathrm{P}_{\text{xy}}(\mathbf{R}_{\text{cam}j} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\text{3dn}ji})^{\mathsf{T}}}{\| \mathbf{S}\, \mathrm{P}_{\text{xy}}(\mathbf{R}_{\text{cam}j} \mathbf{R}^{\mathsf{T}} \hat{\mathbf{y}}_{\text{3dn}ji}) \|} \hat{\mathbf{x}}_{\text{2dn}ji} \right)$$

$$\text{subject to:} \quad s_{\min} \leq s \leq s_{\max}$$

$$\text{where} \quad \mathbf{d}_{\text{sfm}i} = \mathbf{y}_{\text{sfm}i} - s\mathbf{R}\mathbf{x}_{\text{sfm}i} - \mathbf{t}$$

$$\mathbf{d}_{\text{ctr}ji} = f\mathbf{S}\, \Pi(\mathrm{T}_{\text{cam}j}(\frac{1}{s}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{\text{3dp}ji} - \mathbf{t}))) - \mathbf{x}_{\text{2dp}ji}$$

18 Enforce that the camera positions remain within the anatomical constraint:

$\boldsymbol{\Delta}\mathbf{T} = [\boldsymbol{\Delta}\mathbf{s}, \boldsymbol{\Delta}\mathbf{R}, \boldsymbol{\Delta}\mathbf{t}] \leftarrow \mathbf{T} \circ \mathbf{T}_{\text{prev}}^{-1}$

$\boldsymbol{\Delta}\boldsymbol{a} \leftarrow$ Rodrigues parameterization of $\boldsymbol{\Delta}\mathbf{R}$ (Equation 6.7)

$step \leftarrow 1.0$

while *any camera position is outside the anatomical boundary* **do**

$step \leftarrow backupFactor \cdot step$

$\mathbf{R}_{\text{step}} \leftarrow \mathrm{R}(step \cdot \boldsymbol{\Delta}\boldsymbol{a})$

$\mathbf{t}_{\text{step}} \leftarrow \mathrm{R}((1 - step)\boldsymbol{\Delta}\boldsymbol{a}) \cdot (step \cdot \boldsymbol{\Delta}\mathbf{t})$

$s_{\text{step}} \leftarrow step \cdot \boldsymbol{\Delta}\mathbf{s}$

$\mathbf{T} = [s, \mathbf{R}, \mathbf{t}] \leftarrow \mathbf{T}_{\text{step}} \circ \mathbf{T}_{\text{prev}}$ where $\mathbf{T}_{\text{step}} = [s_{\text{step}}, \mathbf{R}_{\text{step}}, \mathbf{t}_{\text{step}}]$

end

end

19 Return the final transformation: $\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$

pairs that are identified as outliers are removed from the set of features that are used when computing the registration phase. Removal of outlying features is signified in Algorithm 8.1 by assigning a weight value of zero to the feature pairs; these weight values are applied to the features within the registration phase of Step 17. Non-outlying features are assigned a weight value of 1, which imparts no change to the feature. The sets of outlier weights are denoted by $\{\varphi_{\text{sfm}i}\}$ for the SFM features and $\{\varphi_{\text{ctr}ji}\}$ for the contour features. Outlier detection can be optionally disabled during the first iteration using the boolean input $\varphi_{\text{disable\_1stIter}}$, which is used in Step 7. Considering all of the features as inliers within the first iteration is often helpful because the initial alignment may by chance place good features both near to and far from the model shape. Performing one iteration with all of the features active can prevent good features from being prematurely cast as outliers by bringing most good features near to the model shape. Disabling outliers during the first iteration is the default setting.

Regarding the SFM features, multiple stages of outlier rejection are employed. The first stage of outlier rejection (Step 8) involves removing the feature pairs whose match errors are in the upper $r_{\text{trim}}\%$ of the SFM match errors. This technique is used due to the high outlier rate that tends to characterize 3D feature locations that are computed using SFM. The percentage of feature pairs that are trimmed is set by the user as an input to the V-IMLOP algorithm. This outlier rejection technique is similar to the method that is used by the Trimmed ICP[128] algorithm and by Mirota

et al.,[24] with a difference being that the V-IMLOP algorithm trims the feature pairs based on the match errors, whereas the prior methods trim the feature pairs based on the match distances. These two approaches are not equivalent if the features are assigned anisotropic uncertainties or if the uncertainties of the features are not all the same. Following the initial trimming of the SFM feature pairs, remaining outliers are identified using a chi-square test, by applying the techniques previously used for the IMLP and G-IMLOP algorithms. In order to account for the overall misalignment of the data shape, the mean square match distance, $\sigma^2_{\text{match\_sfm}}$, is computed (Step 9) and added to the uncertainty of each SFM feature when evaluating the chi-square outlier test in Step 10. The p-value for the chi-square test, p_{sfm}, is an input to the algorithm that is provided by the user, with the default value being 0.95. Note that the chi-square test for the SFM features evaluates the inverse CDF of a chi-square distribution with three degrees of freedom, since the SFM features are distributed in 3D space. Further details regarding this chi-square technique for detecting outliers may be found in the earlier in-depth discussion of Section 4.2 regarding Equation (4.7).

Regarding the contour features, only a chi-square test is used to detect outliers with one difference being that a chi-square test is performed on both the position and orientation components of each feature. In like manner to the SFM features, the mean square match distance and the mean square match angle are first computed (Step 12) in order to account for global shape misalignment and are added to the uncertainty parameters while performing the chi-square tests (Step 13). Note

that chi-square distributions with two and one degree(s) of freedom are used for the position and orientation chi-square tests, respectively, corresponding to the inherent dimensionality of each feature type. The chi-square test for the orientation component is conducted by using the 1D wrapped normal distribution approximation of the von Mises distribution.[85] A similar approximation using the wrapped normal distribution for performing a chi-square test with orientation feature data was described in Section 6.5.1 regarding the G-IMLOP algorithm. Finally, an optional upper limit on the percentage of outliers for each image is enforced in Step 15, with the upper limit being defined by the input $r_{\text{outlier\_max}}$. Since the outlier detection adapts to the current set of inliers, placing an upper limit on the percentage of outliers can be helpful to prevent the algorithm from locking in on only a few contours and regarding all the other contours as outliers. Other techniques for identifying outliers are possible and could be substituted in place of the outlier detection strategies described in this and the prior paragraph.

In the registration phase of Step 17, support is included for optional limits on the estimated scale, which is defined by the user-supplied upper and lower bounds, $s_{\max}$ and $s_{\min}$. Applying constraints to the scale can make the registration more robust by ensuring that an unrealistic scale is not computed in the initial iterations when misalignment of the data and model shapes may be quite large. An initial estimate of the scale could be obtained either by tracking the endoscope using an external tracking system or by making inferences from anatomical structures of known size in

the video images, for example. The scale constraints can then be used to limit the scale within some interval relative to the initial scale estimate.

As previously mentioned, the V-IMLOP algorithm enforces an anatomical constraint on the estimated camera positions. The anatomical constraint is intended to ensure that the estimated camera positions do not exit the feasible region defined by the sinus airway. Not only does this help the algorithm find the correct alignment, but it also protects the algorithm from experiencing what would likely be an irrecoverable fault condition. Because the model-shape contours that are computed during each iteration are defined by the portions of the model surface that are visible from the estimated camera poses, then it is clear that if an estimated camera position moves to the wrong side of the surface mesh that defines the interior sinus wall, then all the image contours that are apparent to that camera will change completely and will share no resemblance with the contours from the video images. It is unlikely that the algorithm would be able to recover from such a condition.

An anatomical constraint prevents this fault condition from happening and can be simply implemented using the framework already provided by the V-IMLOP algorithm. The anatomical constraint is enforced by computing the nearest point on the mesh surface to the optical center of each estimated camera position and examining the direction of the surface normal of the mesh at that location. If the surface normal is oriented towards the optical center of the camera, rather than away from the optical center, then the camera position is valid and is located on the correct side of the

mesh. Otherwise, the camera is located at an invalid position on the opposite side of the mesh. This description assumes that the surface normals of the interior wall of the sinus are oriented towards the sinus airway (i.e., towards the sinus interior).

The anatomical constraints are applied in Step 18 of the algorithm, immediately following the registration phase. In order to keep the registration phase straightforward, the registration phase first computes an updated transformation without regard to the anatomical constraints. Following the registration, the anatomical constraints are examined to determine if any estimated camera position is invalid. If any camera is located at an invalid position, then the updated transformation is modified in order to backup towards the prior transformation by a fractional amount of the motion between the current and prior transformation. This backing up of the transformation continues until all of the cameras have returned to a valid pose. The modified transformation is computed such that the origin of the data shape backs up along a straight path connecting the prior and newly registered transformations; following a straight path requires modifying not only the magnitude but also the direction of the translation vector, $\mathbf{t}_{\text{step}}$, in Step 18.

Using the anatomical constraint requires that all of the camera positions be initialized within the feasible region before beginning the registration; a valid initialization could be achieved either manually or automatically. One possible approach for automatic initialization is to deformably register the pre-operative CT of the patient with a template CT that contains labels identifying all the voxels comprising the sinus

airway. The voxels of the patient CT that map to these labels would then define the feasible region for initializing the camera positions.

In addition to enforcing an anatomical constraint on the optical center positions of the cameras, it is also possible to enforce a similar constraint on the entire shaft of the endoscope, since the endoscope must pass only through the sinus airway (with exception of holes that may be created in the sinus wall during surgery). This can be simply accomplished by seeding points along the endoscope shaft that should be confined to the feasible region, using as many or as few constraining points as desired. While these added constraints are not necessary to prevent the irrecoverable fault condition described above, they may help to determine the correct alignment by further constraining the space of feasible registration solutions.

8.4 Correspondence Phase

This section describes an efficient implementation for the correspondence phase of the V-IMLOP algorithm, wherein the most likely matches are computed from the model shape that correspond to the data-shape features.

8.4.1 Matching the SFM Features

Considering the SFM data-shape features, the matches are computed from the model shape that minimize the match error equation of (8.15), as shown below.

$$\mathbf{y}_{\text{sfm}i} \leftarrow \underset{\mathbf{y}_{\text{sfm}} \in \Psi}{\operatorname{argmin}} \left(\frac{1}{2} (\mathbf{y}_{\text{sfm}} - s\mathbf{R}\mathbf{x}_{\text{sfm}i} - \mathbf{t})^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_{\text{sfm}}^{-1} \mathbf{R}^{\mathsf{T}} (\mathbf{y}_{\text{sfm}} - s\mathbf{R}\mathbf{x}_{\text{sfm}i} - \mathbf{t}) \right) \tag{8.21}$$

The strategy for computing these matches is a straightforward simplification of the correspondence search strategy of the IMLP algorithm that was described earlier in Section 4.3, with the difference being that the V-IMLOP algorithm defines an uncertainty model only on the data-shape features, whereas the IMLP algorithm defines an uncertainty model on both the data and model shape. A similar simplification was described in Section 7.4 regarding the P-IMLOP algorithm. However, it is straightforward to add an uncertainty component to the model shape as well, if desired, since in this case the correspondence search strategy of the IMLP algorithm is applied without simplification. Algorithm 8.2 summarizes this PD-tree search strategy for matching the SFM features.

Note that for a model shape represented by a mesh (rather than by a point cloud) the datum point, $\mathbf{y}_j$, that is used to compute the datum match error in line 5 of Algorithm 8.2 is obtained by setting $\mathbf{y}_j$ to the point on the datum triangle that is the most likely match for $\mathbf{x}_{\text{sfm}}$, which is discussed in Appendix B.

Algorithm 8.2. PD-Tree Node Search for SFM Feature Matching

input : SFM data point to be matched: $\mathbf{x}_{\text{sfm}}$
SFM data uncertainty model: $\mathbf{\Sigma}_{\text{sfm}}$
Current registration parameters: $[s, \mathbf{R}, \mathbf{t}]$
Current candidate for best match and its match error: $[\mathbf{y}_{\text{best}}, E_{\text{best}}]$
Current node being searched: $\mathcal{N}$

output: Updated best match: $[\mathbf{y}_{\text{best}}, E_{\text{best}}]$

Compute an intersection test between the ellipsoid

$$\mathcal{E} = \{\mathbf{z} \mid (\mathbf{z} - s\mathbf{R}\mathbf{x}_{\text{sfm}} - \mathbf{t})^{\mathsf{T}}\mathbf{R}\mathbf{\Sigma}_{\text{sfm}}^{-1}\mathbf{R}^{\mathsf{T}}(\mathbf{z} - s\mathbf{R}\mathbf{x}_{\text{sfm}} - \mathbf{t}) \leq 2E_{\text{best}}\}$$

and the oriented bounding box of the node, $\mathcal{N}.OBB$

if $\mathcal{E}$ *intersects* $\mathcal{N}.OBB$ **then**
  if $\mathcal{N}$ *is a leaf node* **then**
    foreach $datum_j \in \mathcal{N}$ **do**
      Compute the match error for this datum:
        $E_j \leftarrow \text{E}_{\text{V-IMLOP,sfm}}(\mathbf{x}_{\text{sfm}}, \mathbf{y}_j, \mathbf{\Sigma}_{\text{sfm}}, s, \mathbf{R}, \mathbf{t})$ (Equ. (8.15), App. B)
      if $E_j < E_{best}$ **then** update the best match:
        $[\mathbf{y}_{\text{best}}, E_{\text{best}}] \leftarrow [\mathbf{y}_j, E_j]$
    end
  else
    Search the left and right child nodes of $\mathcal{N}$
  end
end
Return the updated best match: $[\mathbf{y}_{\text{best}}, E_{\text{best}}]$

8.4.2 Matching the Contour Features

Computing the matches for the video contour features consists of a two-stage process, as described earlier in Section 8.2. In the first stage, the points on the model shape are computed that produce visible occluding contours after being projected to the image planes. In the second stage, matches for the video contours are computed from the set of projected model-shape contours that are associated with the same video image.

8.4.2.1 Stage 1: Computing the Model-Shape Contours

This section describes an implementation for computing the occluding contours of the model shape from a given camera viewpoint. This discussion is limited to occluding contours for a model shape represented by a triangular surface mesh. For the purposes of this discussion, the direction of the ray that passes from the optical center of a camera to some point on the face of a given triangle in the mesh is denoted by $\hat{\mathbf{r}}_{\text{cam}}$ and the face normal of the same triangle is denoted by $\hat{\mathbf{n}}_{\text{face}}$.

The key idea is based on the simple fact that all occluding contours of the model that are visible within a projected image are formed from the projections of select triangle edges, where the face of one triangle sharing the edge is visible to the camera (i.e., $\hat{\mathbf{n}}_{\text{face}}^{\mathsf{T}}\hat{\mathbf{r}}_{\text{cam}} < 0$) and the face of the other triangle sharing the edge is not visible to the camera (i.e., $\hat{\mathbf{n}}_{\text{face}}^{\mathsf{T}}\hat{\mathbf{r}}_{\text{cam}} \geq 0$). Note that only two triangles may share the same edge. Thus, all occluding contours for a given camera viewpoint may be found by

searching through all the triangle faces; for each triangle face that is visible to the camera, check if any neighboring triangle is not visible. If a neighbor triangle is not visible, then the edge that is shared with that neighbor is added to the set of occluding model-shape contours.

However, it is not sufficient to simply find all the triangle edges of the model that create occluding contours in this manner; what is needed are all the triangle edges of the model that create *visible* occluding contours within the image. In other words, many of the occluding contours found by following the procedure of the prior paragraph could be occluded by other regions of the model surface that lie in between the image plane and the 3D edge that projects to an occluding contour. In order to identify only the visible occluding contours, the technique of Z-buffering may be used.[130] This technique involves first rendering the model shape in order to compute the depth of the nearest point on the model surface that projects to each pixel in the video image; these depths are stored in what is called the Z-buffer. Note that this rendering should be performed without culling. Following this initial rendering, the occluding contours of the model shape are then computed and their depths are compared to the corresponding pixel location within the Z-buffer. If the depth of a contour point is less than or equal to the depth in the Z-buffer, then that contour point must be visible and is included in the set of visible contours.

A GPU-based implementation of this approach for computing the model-shape contours was programmed in C++ using the DirectX graphics libraries. In practice,

additional calculations are needed to deal with problems such as self-shadowing, which can result in visible contours being excluded due to numerical errors in the depth calculations.[130] The approach used within the author's implementation in order to deal with the problem of self-shadowing was to add a small additive depth bias to the first-pass rendering, causing the model shape to be rendered at slightly higher depth. This additive bias makes it far less likely for numerical errors to result in visible contours being shadowed by the initial rendering.

Since the occluding contours are formed from a set of triangle edges, the output of the DirectX program is a set of vertices defining the two end points of the triangle edges that produce the visible contours. Both the 3D locations of each vertex on the model shape and the 2D locations of each vertex projected to the video image are output from the program, which provides both the 2D and 3D contour locations as required by the correspondence and registration phases, respectively, of the V-IMLOP algorithm. In addition, normal orientations for the edges are also computed. Note that the normal orientations of these edges in 3D space are not unique, however, and could be interpreted to be any orientation in between the face normals of the two triangles forming each edge. In other words, the face normal of the visible triangle provides one possible extreme orientation for the edge and the face normal of the non-visible triangle provides another possible extreme orientation for the edge. In addition, all intermediate face normal orientations that are encountered while rotating the first triangle about the edge until its face normal aligns with the second face

normal may also be considered as valid edge orientations, with the direction of rotation being in the direction pointed by the visible face normal. Since for this application we are interested in the orientation of the edge when projected to the image plane, the 3D edge orientation is defined to be the orientation within the range of valid edge orientations that is parallel to the imaging plane. This orientation is unique and always exists. This choice of edge orientation underscores the importance of using the orthographic projection, P_{xy} (Equation (8.14)), rather than the perspective projection, Π (Equation (8.6)), in order to avoid division by zero when projecting the 3D edge orientation onto the image plane, as discussed earlier in Section 8.2.

8.4.2.2 Stage 2: Matching

Matches for the video contour features are computed from the set of estimated model-shape contours that were computed for the same image. The set of model-shape contours for a given video image are computed by using the current registration value to estimate the camera pose associated with the video image and then applying the technique for computing the model-shape contours from that camera pose, as described in the prior section. A match for a given video contour feature is then computed by minimizing the match error equation of (8.17). In like manner with the prior chapters, the following alternative match error function is used for implementing

the correspondence search

$$\mathrm{E}_{\text{V-IMLOP,2d}}(\mathbf{x}_{\text{2d}}, \mathbf{y}_{\text{2d}}, \boldsymbol{\Sigma}_{\text{2d}}, \kappa) = \frac{1}{2}(\mathbf{y}_{\text{2dp}} - \mathbf{x}_{\text{2dp}})^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{2d}}^{-1} (\mathbf{y}_{\text{2dp}} - \mathbf{x}_{\text{2dp}}) + \kappa(1 - \hat{\mathbf{y}}_{\text{2dn}}^{\mathsf{T}} \hat{\mathbf{x}}_{\text{2dn}}) \ . \quad (8.22)$$

which, unlike (8.17), is always positive by addition of an extra κ term. The match for the ith data-shape contour feature ($\mathbf{x}_{\text{2d}ji} = (\mathbf{x}_{\text{2dp}ji}, \hat{\mathbf{x}}_{\text{2dn}ji})$) belonging to the jth image is therefore computed by minimizing the match error equation of (8.22), as shown below.

$$(\mathbf{y}_{\text{2dp}ji}, \hat{\mathbf{y}}_{\text{2dn}ji}) = \underset{(\mathbf{y}_{\text{2dp}}, \hat{\mathbf{y}}_{\text{2dn}}) \in \Psi_j}{\operatorname{argmin}} \left(\frac{1}{2}(\mathbf{y}_{\text{2dp}} - \mathbf{x}_{\text{2dp}ji})^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{2d}ji}^{-1} (\mathbf{y}_{\text{2dp}} - \mathbf{x}_{\text{2dp}ji}) + \kappa_{ji}(1 - \hat{\mathbf{y}}_{\text{2dn}}^{\mathsf{T}} \hat{\mathbf{x}}_{\text{2dn}ji}) \right) \quad (8.23)$$

Since the match error equation of (8.22) combines a 2D *anisotropic* distribution of a position feature with a 2D isotropic distribution of an orientation feature, the correspondence search may be implemented as a straightforward simplification of the PD-tree search strategy that was developed for the G-IMLOP algorithm, which is described in Section 6.3. The simplifications include performing the search in 2D rather than 3D space and using an isotropic rather than an anisotropic distribution for the orientation. Note that the probability distribution for the 2D orientations is isotropic because only one degree of freedom differentiates orientations on the 2D

unit circle.

The contour features for a given video image are matched to the projected model-shape contours that are associated with that video image by searching a PD tree that is constructed around the projected model-shape contours. Thus, an independent PD tree is constructed from the projected model-shape contours that are associated with each video image. Since the estimated camera positions, and thus the projected model-shape contours, change during each iteration, these PD trees are recomputed at the beginning of each correspondence phase.

Recall that in order to apply the PD-tree search technique that is used by the G-IMLOP algorithm (Section 6.3), it is required to establish a lower bound on the orientation component of the match error in order to test whether to search a given node of the PD tree. As a consequence of V-IMLOP's *isotropic* uncertainty regarding the orientation data, the lower bound for a node's orientation error may be established using the more simple isotropic technique of the IMLOP algorithm (Section 5.3). Assuming that the average datum orientation, $\hat{\mathbf{n}}_{\text{avg}}$, and the maximum angular deviation from the average datum orientation, $\theta_{\max}$, are stored as parameters of each node, then a lower bound, $\theta_{\min}$, on the minimum angular error between the data orientation, $\hat{\mathbf{x}}_{2\text{dn}}$, and any model orientation, $\{\hat{\mathbf{y}}_{2\text{dn}i}\}$, contained within the node may be computed as

$$\theta_{\min} = \max\left(0\,,\, \operatorname{acos}\left(\hat{\mathbf{n}}_{\text{avg}}^{\mathsf{T}}\hat{\mathbf{x}}_{2\text{dn}}\right) - \theta_{\max}\right) \quad . \tag{8.24}$$

Note that the registration parameters, $[s, \mathbf{R}, \mathbf{t}]$, do not appear in Equation (8.24), since the registration was already accounted for by the projected model-shape contours prior to constructing the PD tree. Thus, the model-shape orientations, $\{\hat{\mathbf{y}}_{2\text{dn}i}\}$, from which $\hat{\mathbf{n}}_{\text{avg}}$ is computed, are already defined in the same coordinates as the data-shape orientation, $\hat{\mathbf{x}}_{2\text{dn}}$. Given the lower bound, $\theta_{\min}$, on the magnitude of angular error, a tight lower bound ($E_{\text{n,min}}$) on the orientation component of the match error of Equation (8.22) for the entire node is then given by

$$E_{\text{n,min}} = \kappa(1 - \cos(\theta_{\min})) \ . \tag{8.25}$$

The remaining elements of the PD-tree search procedure follow the techniques described for the G-IMLOP algorithm in Section 6.3, with a straightforward modification being to apply these techniques in 2D rather than 3D. Algorithm 8.3 summarizes the PD-tree node search strategy for matching the video contour features.

Note that the datum point, $\mathbf{y}_{2\text{d}j} = (\mathbf{y}_{2\text{dp}j}, \hat{\mathbf{y}}_{2\text{dn}j})$, that is used to compute the datum match error in line 7 of Algorithm 8.3 is obtained by setting $\hat{\mathbf{y}}_{2\text{dn}j}$ to the datum edge normal and $\mathbf{y}_{2\text{dp}j}$ to the point on the datum edge that is the most likely match by position from $\mathbf{x}_{2\text{dp}}$, which is a straightforward simplification of the discussion in Appendix B from a 3D case of triangle matching to a 2D case of edge matching.

Algorithm 8.3. PD-Tree Node Search for 2D Contour Feature Matching

input : Oriented contour feature being matched: $\mathbf{x}_{2\mathrm{d}} = (\mathbf{x}_{2\mathrm{dp}}, \hat{\mathbf{x}}_{2\mathrm{dn}})$
Contour feature uncertainty model: $\mathbf{\Sigma}_{2\mathrm{d}}, \kappa$
Current candidate for best match and its match error: $[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}]$
Node being searched: $\mathcal{N}$

output: Updated best match: $[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}]$

Compute $\theta_{\min}$ for this node using $\mathcal{N}.\hat{\mathbf{n}}_{\mathrm{avg}}$ and $\mathcal{N}.\theta_{\max}$ (Equ. 8.24)
Compute a lower bound, $E_{\mathrm{n,min}}$, for the orientation component of the match error for this node using $\theta_{\min}$ (Equ. 8.25)
Compute an intersection test between the ellipsoid

$$\mathcal{E} = \{\mathbf{z} \mid (\mathbf{z} - \mathbf{x}_{2\mathrm{dp}})^{\mathsf{T}} \mathbf{\Sigma}_{2\mathrm{d}}^{-1} (\mathbf{z} - \mathbf{x}_{2\mathrm{dp}}) \leq 2(E_{\mathrm{best}} - E_{\mathrm{n,min}})\}$$

and the oriented bounding box of the node, $\mathcal{N}.OBB$
if $\mathcal{E}$ *intersects* $\mathcal{N}.OBB$ **then**
  if $\mathcal{N}$ *is a leaf node* **then**
    foreach $datum_j \in \mathcal{N}$ **do**
      Compute the match error for this datum:
        $E_j \leftarrow \mathrm{E}_{\text{V-IMLOP,2d}}(\mathbf{x}_{2\mathrm{d}}, \mathbf{y}_{2\mathrm{d}j}, \mathbf{\Sigma}_{2\mathrm{d}}, \kappa)$ (Equ. (8.17))
      if $E_j < E_{best}$ **then** update the best match:
        $[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}] \leftarrow [\mathbf{y}_{2\mathrm{d}j}, E_j]$
    end
  else
    Search the left and right child nodes of $\mathcal{N}$
  end
end
Return the updated best match: $[\mathbf{y}_{\mathrm{best}}, E_{\mathrm{best}}]$

8.5 Registration Phase

This section describes an implementation for the registration phase of the V-IMLOP algorithm, wherein the SFM and video contour features ($\mathbf{X}_{\text{sfm}} = \{\mathbf{x}_{\text{sfm}i}\}$ and $\mathbf{X}_{\text{ctr}} = \{(\mathbf{x}_{\text{2dp}ji}, \hat{\mathbf{x}}_{\text{2dn}ji})\}$, respectively) of the data shape are registered with the sets of matching model-shape features ($\mathbf{Y}_{\text{sfm}} = \{\mathbf{y}_{\text{sfm}i}\}$ and $\mathbf{Y}_{\text{ctr}} = \{(\mathbf{y}_{\text{3dp}ji}, \hat{\mathbf{y}}_{\text{3dn}ji})\}$, respectively) that have been computed from the preceding correspondence phase. The goal is to compute the similarity transformation, $\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$, that minimizes the total match error of Equation (8.20), which is copied below where the optional constraint on the scale value has been included.

$$\mathbf{T} = \underset{[s,\mathbf{R},\mathbf{t}]}{\text{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n_{\text{sfm}}} \mathbf{d}_{\text{sfm}i}^{\mathsf{T}} \mathbf{R} \boldsymbol{\Sigma}_{\text{sfm}i}^{-1} \mathbf{R}^{\mathsf{T}} \mathbf{d}_{\text{sfm}i} + \frac{1}{2} \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} \mathbf{d}_{\text{ctr}ji}^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{2d}ji}^{-1} \mathbf{d}_{\text{ctr}ji} - \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} \kappa_{ji} \frac{\mathbf{S}\,\mathrm{P}_{\text{xy}}(\mathbf{R}_{\text{cam}j}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{\text{3dn}ji})^{\mathsf{T}}}{\|\mathbf{S}\,\mathrm{P}_{\text{xy}}(\mathbf{R}_{\text{cam}j}\mathbf{R}^{\mathsf{T}}\hat{\mathbf{y}}_{\text{3dn}ji})\|} \hat{\mathbf{x}}_{\text{2dn}ji} \right)$$

$$\text{subject to:} \quad s_{\min} < s < s_{\max}$$

$$\mathbf{d}_{\text{sfm}i} = \mathbf{y}_{\text{sfm}i} - s\mathbf{R}\mathbf{x}_{\text{sfm}i} - \mathbf{t}$$

$$\mathbf{d}_{\text{ctr}ji} = f\mathbf{S}\,\Pi(\mathrm{T}_{\text{cam}j}(\frac{1}{s}\mathbf{R}^{\mathsf{T}}(\mathbf{y}_{\text{3dp}ji} - \mathbf{t}))) - \mathbf{x}_{\text{2dp}ji}$$

Due to the non-linearity of this objective function, an iterative method is required for its optimization. In the author's implementation, Matlab's optimization toolbox is used for this purpose. If constraints on the scale are provided by the user, then Matlab's quasi-Newton-based active-set method for constrained optimization is used;

otherwise, Matlab's unconstrained quasi-Newton method is employed. In both cases, the gradient of the objective function is computed analytically using the gradient equations that are discussed below.

As described for earlier algorithms, in order to properly minimize the objective of (8.20), the constraints that characterize a valid rotation matrix must be maintained, namely $\mathbf{R}^\mathsf{T}\mathbf{R} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$. As done for earlier algorithms, these constraints are enforced by re-parameterizing the rotation matrix, $\mathbf{R}$, using the Rodrigues form, $\boldsymbol{a} = [\alpha_x, \alpha_y, \alpha_z]^\mathsf{T}$. Recall that the notation $\mathrm{R}(\boldsymbol{a})$ signifies the 3×3 rotation matrix that corresponds to the Rodrigues vector, $\boldsymbol{a}$, as defined in Equation (6.7).

The quasi-Newton optimizers that are employed require having the gradient of the objective function with respect to the transformation parameters, $[s, \mathbf{R}, \mathbf{t}]$. Incorporating the Rodrigues parameterization of the rotation matrix along with some reformatting in order to simplify forming the gradient expressions produces Equa-

tion (8.26) as the objective function that is used for forming the gradient expressions.

$$\mathbf{T} = \underset{[s,\boldsymbol{a},\mathbf{t}]}{\operatorname{argmin}} \left(\sum_{i=1}^{n_{\text{sfm}}} C_{\text{sfm}i} + \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} C_{\text{ctrp}ji} + \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} C_{\text{ctrn}ji} \right) \tag{8.26}$$

$$\text{subject to:} \quad s_{\min} < s < s_{\max}$$

$$C_{\text{sfm}i} = \frac{1}{2} \mathbf{z}_{\text{sfm}i}^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{sfm}i}^{-1} \mathbf{z}_{\text{sfm}i}$$

$$\mathbf{z}_{\text{sfm}i} = \mathrm{R}(\boldsymbol{a})^{\mathsf{T}} (\mathbf{y}_{\text{sfm}i} - \mathbf{t}) - s\mathbf{x}_{\text{sfm}i}$$

$$C_{\text{ctrp}ji} = \frac{1}{2} \mathbf{z}_{\text{ctr}ji}^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{2d}ji}^{-1} \mathbf{z}_{\text{ctr}ji}$$

$$\mathbf{z}_{\text{ctr}ji} = f\mathbf{S}\,\Pi(\mathbf{R}_{\text{cam}j}\,\mathrm{R}(\boldsymbol{a})^{\mathsf{T}}(\mathbf{y}_{\text{3dp}ji} - \mathbf{t}) + s\mathbf{t}_{\text{cam}j}) - \mathbf{x}_{\text{2dp}ji}$$

$$C_{\text{ctrn}ji} = -\kappa_{ji} \frac{\mathbf{S}\,\mathrm{P}_{\text{xy}}(\mathbf{R}_{\text{cam}j}\,\mathrm{R}(\boldsymbol{a})^{\mathsf{T}} \hat{\mathbf{y}}_{\text{3dn}ji})^{\mathsf{T}}}{\|\mathbf{S}\,\mathrm{P}_{\text{xy}}(\mathbf{R}_{\text{cam}j}\,\mathrm{R}(\boldsymbol{a})^{\mathsf{T}} \hat{\mathbf{y}}_{\text{3dn}ji})\|} \hat{\mathbf{x}}_{\text{2dn}ji}$$

Equations for the gradient ($\nabla\mathbf{C}$) of the objective function of (8.26) with respect to the transformation parameters, s, $\boldsymbol{a}$, and $\mathbf{t}$, are provided below, where the notation $\mathbf{J}_{a,b}$ signifies the Jacobian of an expression, a, with respect to some variable expression, b.

$$\nabla \mathbf{C} = \sum_{i=1}^{n_{\text{sfm}}} \nabla \mathbf{C}_{\text{sfm}i} + \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} \nabla \mathbf{C}_{\text{ctrp}ji} - \sum_{j=1}^{n_{\text{cam}}} \sum_{i=1}^{n_{\text{ctr}j}} \nabla \mathbf{C}_{\text{ctrn}ji} \tag{8.27}$$

$$\nabla \mathbf{C}_{\text{sfm}i} = \left[\mathbf{J}_{C_{\text{sfm}i},\mathbf{z}_{\text{sfm}i}} \mathbf{J}_{\mathbf{z}_{\text{sfm}i},\boldsymbol{a}}, \quad \mathbf{J}_{C_{\text{sfm}i},\mathbf{z}_{\text{sfm}i}} \mathbf{J}_{\mathbf{z}_{\text{sfm}i},\mathbf{t}}, \quad \mathbf{J}_{C_{\text{sfm}i},\mathbf{z}_{\text{sfm}i}} \mathbf{J}_{\mathbf{z}_{\text{sfm}i},s} \right]^{\mathsf{T}} \tag{8.28}$$

$$\mathbf{J}_{C_{\text{sfm}i},\mathbf{z}_{\text{sfm}i}} = \mathbf{z}_{\text{sfm}i}^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{sfm}i}^{-1} \tag{8.29}$$

$$\mathbf{J}_{\mathbf{z}_{\text{sfm}i},\boldsymbol{a}} = \left[\frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_x}^{\mathsf{T}} (\mathbf{y}_{\text{sfm}i} - \mathbf{t}), \quad \frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_y}^{\mathsf{T}} (\mathbf{y}_{\text{sfm}i} - \mathbf{t}), \quad \frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_z}^{\mathsf{T}} (\mathbf{y}_{\text{sfm}i} - \mathbf{t}) \right] \tag{8.30}$$

$$\mathbf{J}_{\mathbf{z}_{\text{sfm}i},\mathbf{t}} = -\,\mathrm{R}(\boldsymbol{a})^{\mathsf{T}} \tag{8.31}$$

$$\mathbf{J}_{\mathbf{z}_{\text{sfm}i},s} = -\mathbf{x}_{\text{sfm}i} \tag{8.32}$$

$$\begin{aligned} \nabla \mathbf{C}_{\text{ctrp}ji} = \Big[& \mathbf{J}_{C_{\text{ctrp}ji},\mathbf{z}_{\text{ctr}ji}} \mathbf{J}_{\mathbf{z}_{\text{ctr}ji},\mathbf{y}_{\text{xfmp}ji}} \mathbf{J}_{\mathbf{y}_{\text{xfmp}ji},\boldsymbol{a}}, \\ & \mathbf{J}_{C_{\text{ctrp}ji},\mathbf{z}_{\text{ctr}ji}} \mathbf{J}_{\mathbf{z}_{\text{ctr}ji},\mathbf{y}_{\text{xfmp}ji}} \mathbf{J}_{\mathbf{y}_{\text{xfmp}ji},\mathbf{t}}, \quad \mathbf{J}_{C_{\text{ctrp}ji},\mathbf{z}_{\text{ctr}ji}} \mathbf{J}_{\mathbf{z}_{\text{ctr}ji},\mathbf{y}_{\text{xfmp}ji}} \mathbf{J}_{\mathbf{y}_{\text{xfmp}ji},s} \Big]^{\mathsf{T}} \end{aligned} \tag{8.33}$$

$$\mathbf{y}_{\text{xfmp}ji} = \mathbf{R}_{\text{cam}j} \, \mathrm{R}(\boldsymbol{a})^{\mathsf{T}} (\mathbf{y}_{\text{3dp}ji} - \mathbf{t}) + s \mathbf{t}_{\text{cam}j} \tag{8.34}$$

$$\mathbf{J}_{C_{\text{ctrp}ji},\mathbf{z}_{\text{ctr}ji}} = \mathbf{z}_{\text{ctr}ji}^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{2d}ji}^{-1} \tag{8.35}$$

$$\mathbf{J}_{\mathbf{z}_{\text{ctr}ji},\mathbf{y}_{\text{xfmp}ji}} = f \mathbf{S} \, \mathrm{J}_{\text{PP}}(\mathbf{y}_{\text{xfmp}ji}) \tag{8.36}$$

$$\mathrm{J}_{\text{PP}}([x,\, y,\, z]^{\mathsf{T}}) = \begin{bmatrix} 1/z & 0 & -x/z^2 \\ 0 & 1/z & -y/z^2 \end{bmatrix} \tag{8.37}$$

$$\begin{aligned} \mathbf{J}_{\mathbf{y}_{\text{xfmp}ji},\boldsymbol{a}} = \mathbf{R}_{\text{cam}j} \Big[& \frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_x}^{\mathsf{T}} (\mathbf{y}_{\text{3dp}ji} - \mathbf{t}), \quad \frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_y}^{\mathsf{T}} (\mathbf{y}_{\text{3dp}ji} - \mathbf{t}), \\ & \frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_z}^{\mathsf{T}} (\mathbf{y}_{\text{3dp}ji} - \mathbf{t}) \Big] \end{aligned} \tag{8.38}$$

$$\mathbf{J}_{\mathbf{y}_{\text{xfmp}ji},\mathbf{t}} = -\mathbf{R}_{\text{cam}j} \, \mathrm{R}(\boldsymbol{a})^{\mathsf{T}} \tag{8.39}$$

$$\mathbf{J}_{\mathbf{y}_{\text{xfmp}ji},s} = \mathbf{t}_{\text{cam}j} \tag{8.40}$$

$$\nabla \mathbf{C}_{\mathrm{ctrn}ji} = \left[\mathbf{J}_{C_{\mathrm{ctrn}ji}, \mathbf{z}_{\mathrm{ctr}ji}} \mathbf{J}_{\mathbf{z}_{\mathrm{ctr}ji}, \mathbf{y}_{\mathrm{xfmn}ji}} \mathbf{J}_{\mathbf{y}_{\mathrm{xfmn}ji}, \boldsymbol{a}}, \quad 0, \quad 0 \right] \tag{8.41}$$

$$\mathbf{y}_{\mathrm{xfmn}ji} = \mathbf{R}_{\mathrm{cam}j} \, \mathrm{R}(\boldsymbol{a})^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{3dn}ji} \tag{8.42}$$

$$\mathbf{y}_{\mathrm{prjn}ji} = \mathbf{S} \, \mathrm{P}_{\mathrm{xy}}(\mathbf{y}_{\mathrm{xfmn}ji}) \tag{8.43}$$

$$\mathbf{J}_{C_{\mathrm{ctrn}ji}, \mathbf{y}_{\mathrm{prjn}ji}} = -\kappa_{ji} \, \hat{\mathbf{x}}_{\mathrm{2dn}ji}^{\mathsf{T}} \left(\frac{1}{\|\mathbf{y}_{\mathrm{prjn}ji}\|_2} \mathbf{I} - \frac{\mathbf{y}_{\mathrm{prjn}ji} \mathbf{y}_{\mathrm{prjn}ji}^{\mathsf{T}}}{\|\mathbf{y}_{\mathrm{prjn}ji}\|_2^3} \right) \tag{8.44}$$

$$\mathbf{J}_{\mathbf{y}_{\mathrm{prjn}ji}, \mathbf{y}_{\mathrm{xfmn}ji}} = \mathbf{S} \, \mathbf{J}_{\mathrm{P_{xy}}} = \mathbf{S} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{8.45}$$

$$\mathbf{J}_{\mathbf{y}_{\mathrm{xfmn}ji}, \boldsymbol{a}} = \mathbf{R}_{\mathrm{cam}j} \left[\frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_x}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{3dn}ji}, \quad \frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_y}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{3dn}ji}, \quad \frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_z}^{\mathsf{T}} \hat{\mathbf{y}}_{\mathrm{3dn}ji} \right] \tag{8.46}$$

In the foregoing equations, $\frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_x}$ signifies the 3×3 matrix of partial derivatives of $\mathrm{R}(\boldsymbol{a})$ with respect to element a_x of $\boldsymbol{a} = [a_x, \, a_y, \, a_z]^{\mathsf{T}}$, and so on for $\frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_y}$ and $\frac{\partial \, \mathrm{R}(\boldsymbol{a})}{\partial a_z}$. The definitions for these partial derivatives are described in Appendix F. Note that Equation (8.37) defines the Jacobian of the perspective projection as the output of an operator, $\mathrm{J_{PP}}(\cdot)$, whereas the Jacobian of the orthographic projection is defined using a simple static matrix, $\mathbf{J}_{\mathrm{P_{xy}}}$, as indicated in Equation (8.45).

8.6 Results

In this section, an experimental evaluation of the V-IMLOP algorithm is presented. Comparison is made to the Trimmed ICP algorithm with scale estimation as described by Mirota, et al.,[127] with one difference being that the rendering-based technique for cropping the model-shape mesh to the regions that are visible from

the estimated camera positions was not incorporated in the implementation. If implemented, this technique would be equally applicable to both the V-IMLOP and Trimmed ICP algorithms, however.

8.6.1 Experiment 1: Registering Patient Data with Real Video Features

This experiment investigates a registration between video and CT data of an anonymized patient obtained under an IRB-approved clinical study at Johns Hopkins Hospital. In this experiment, a short segment of an endoscopic video recording of a patient undergoing endonasal surgery is registered to preoperative CT data of the same patient. The six-frame video segment views the middle turbinate of the patient's left sinus; these images are shown in Figure 8.4. A surface mesh of the interior walls of the sinus was segmented from the preoperative CT data to serve as a model shape for the registration. 3D scaled feature points and the relative extrinsic camera parameters were computed from the sequence of video images using SFM and a SIFT-based feature matching technique. The SFM feature set for this study consisted of 251 points. The 2D video contour features were manually segmented from each video frame shown in Figure 8.4 by fitting a smoothing spline to points marked along each contour edge. The smoothing spline was then sampled at 15-pixel intervals, providing a total of 1158 oriented-point contour features from the six images. To

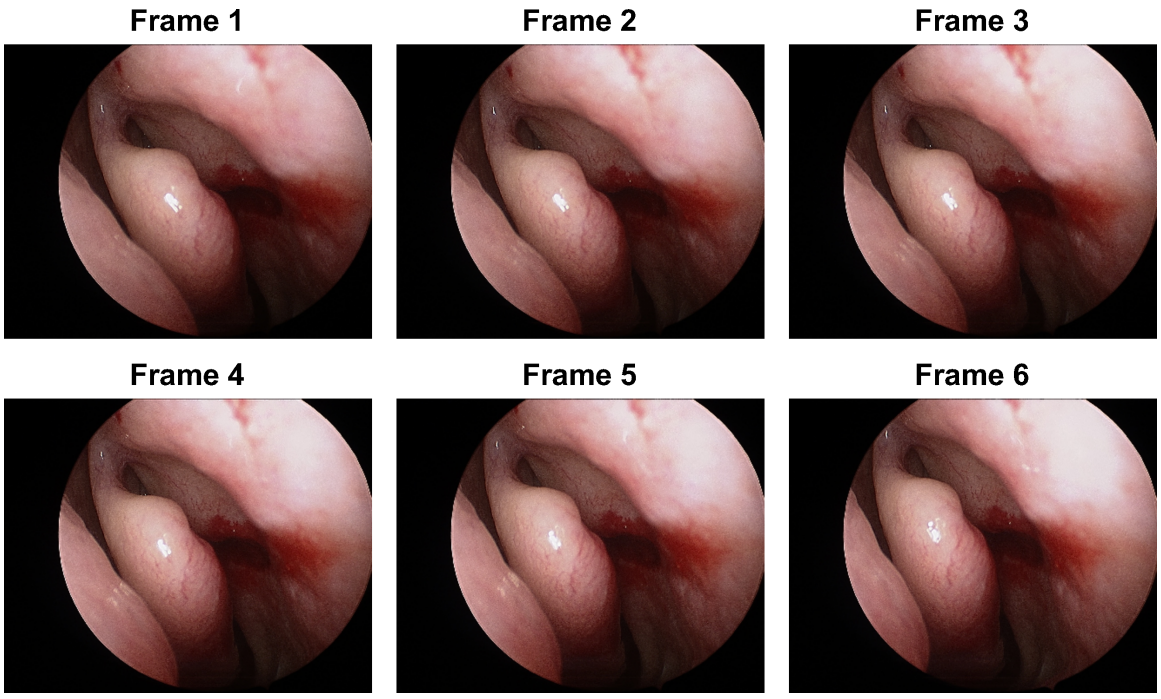

Figure 8.4: Experiment 1: a six-frame segment of endoscopic video recorded from a patient undergoing endonasal surgery.

initialize the registration, the 3D feature points obtained from the SFM procedure were approximately aligned to the mesh of the interior sinus via a manual alignment performed using Blender (www.blender.org).

Both the V-IMLOP and Trimmed ICP algorithms were run using the same initial data alignment and, where applicable, the same settings. The V-IMLOP and Trimmed ICP algorithms were configured to trim 65% of the 3D SFM feature points (i.e., $r_{\text{trim}} = 0.65$). Both algorithms were configured to terminate when the change in the transformation parameters fell below 0.01 degrees rotation, 0.01 mm translation, and 0.001% of the initial scale. Since no measurement-based estimate for the scale was available for this data, no constraints on the scale estimation were applied. The

data covariances ($\{\mathbf{\Sigma}_{\text{sfm}i}\}$) for the SFM feature points were set to 0.5 mm standard deviation in the camera view direction (based on the camera orientation for one of the middle video frames) and 0.3 mm standard deviation in the image parallel directions; this setting is intended to account for the higher uncertainty in the depth direction for video-based reconstruction.[53, 54] These covariances were then multiplied by a scale factor to normalize the relative influences of the SFM and contour feature data (discussed in the following paragraph). The data uncertainty parameters for the contour features were set to isotropic standard deviations of 3 pixels for the position covariances ($\{\mathbf{\Sigma}_{2\text{d}ji}\}$) and an orientation concentration of $\{\kappa_{ji}\} = 200$. The 3 pixels of standard deviation for the contour positions account for segmentation error along with added uncertainty for estimation error in the camera poses. For the contour features, the maximum match angle ($\theta_{\text{match\_max}}$) was set to 30 degrees and the maximum percentage of outliers per image was set to 15% ($r_{\text{outlier\_max}} = 0.15$). Other parameters, such as the p-values (p_{sfm}, p_{ctrp}, p_{ctrn}) for the chi-square tests, etc., were assigned their default settings.

As mentioned above, a scale factor was applied to the covariances of the SFM features in order to normalize the relative influences of the SFM and contour features. This normalization is intended to account for inequality in the number of features that comprise each data set. In particular, since the 2D video contour features far outnumber the 3D SFM features, the contour features will tend to override the influence of the SFM features. This relative influence is even affected by the density at

which the video contours are sampled. From an intuitive point of view, the contour features taken as a whole should not become "more believable" as a result of being sampled at a higher density; however, without normalization this is the effective outcome, since increasing the number of contours diminishes the influence of the SFM data, thereby assigning more "belief" to the contour data. To overcome this obstacle, the SFM covariances are multiplied by the ratio of the number of untrimmed SFM features (i.e., the outlier trim ratio, r_{trim}, for the SFM features is accounted for in the normalization) divided by the total number of contour features. In this manner, the number of contour features can be varied, and the effective influence of the SFM data features will remain unchanged. The ratio described above grants equal influence to the SFM and contour features; however, it is also possible to use different ratios in order to assign relatively more or less influence to one feature type over the other, depending on the user's level of confidence in each feature set.

A significant challenge when evaluating the performance of registration algorithms within the context of clinical patient data concerns how to evaluate the accuracy of the registration outcomes. In general, establishing a ground truth for the registration is particularly challenging, since fiducial-based techniques, which are readily applied in phantom-based studies, cannot be so readily used with a patient due to the additional invasiveness and/or radiation exposure that is typically associated with fiducial-based techniques. Since no registration ground truth was available for this data, the registration outcomes for this study are evaluated by visual inspection of

the final registration outcomes.

Figure 8.5 shows a Blender view of the initial alignment of the 3D SFM points and the initial poses of the cameras with respect to the mesh-based surface model of the sinus that was segmented from preoperative CT. Blender views of the final alignment after registering with the Trimmed ICP and V-IMLOP algorithms are shown in Figures 8.6 and 8.7, respectively. In addition to the blender views, images showing the projections of the estimated model-shape contours onto each video image at the final alignment are shown in Figures 8.8 and 8.9 for the registrations computed by the Trimmed ICP and V-IMLOP algorithms, respectively. In these figures, the projected contours from the 3D model shape, as computed by the DirectX contour estimation software, are shown as green overlays on top of each video image. The manually segmented video contours, which were registered to the projected contours by the V-IMLOP algorithm, are also shown in these figures as white overlays. Note that the Trimmed ICP algorithm did not use these contours during the registration—the contour projections are shown for the Trimmed ICP algorithm only in the interest of investigating the quality of the camera pose alignment that were computed by the Trimmed ICP algorithm. The data-shape scale parameter at initialization was set to 8. The final data-shape scale parameter values computed by each registration using the Trimmed ICP and V-IMLOP algorithms were remarkably close in value at 6.25 and 6.28, respectively.

The initial alignment of Figure 8.5 shows that the majority of the SFM feature

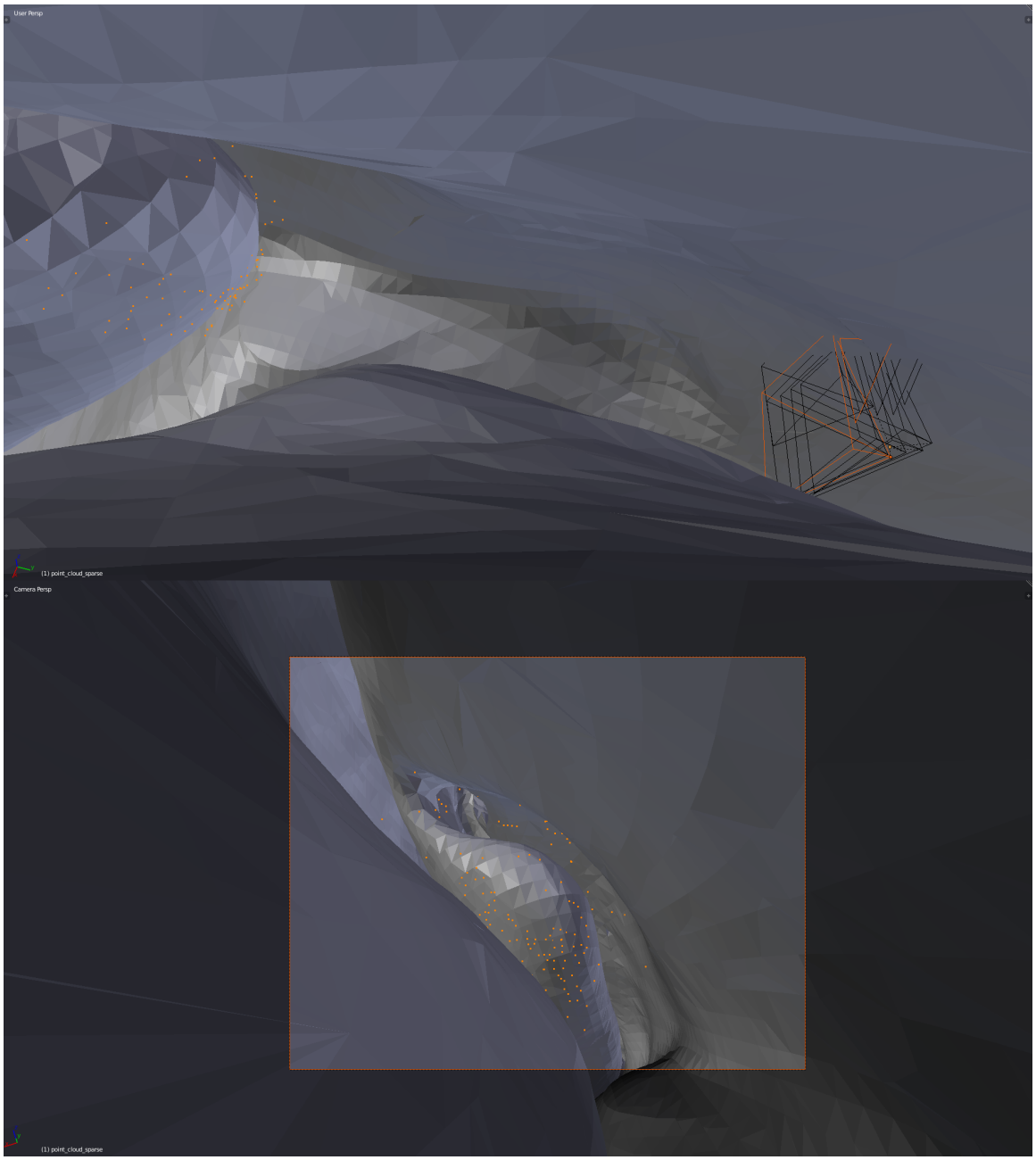

Figure 8.5: Experiment 1: the initial alignment of the SFM feature points and of the camera poses prior to registering the clinical video data using the V-IMLOP and Trimmed ICP algorithms. The upper portion of the figure shows a view from above and looking towards the middle turbinate with the entry to the patient's left nostril being located down the airway to the right of the image. The lower portion of the figure shows the view from the front camera, whose pose is highlighted in the upper figure. The 3D SFM feature points are also highlighted in orange in both the upper and lower figures, shown concentrated near the surface of the middle turbinate.

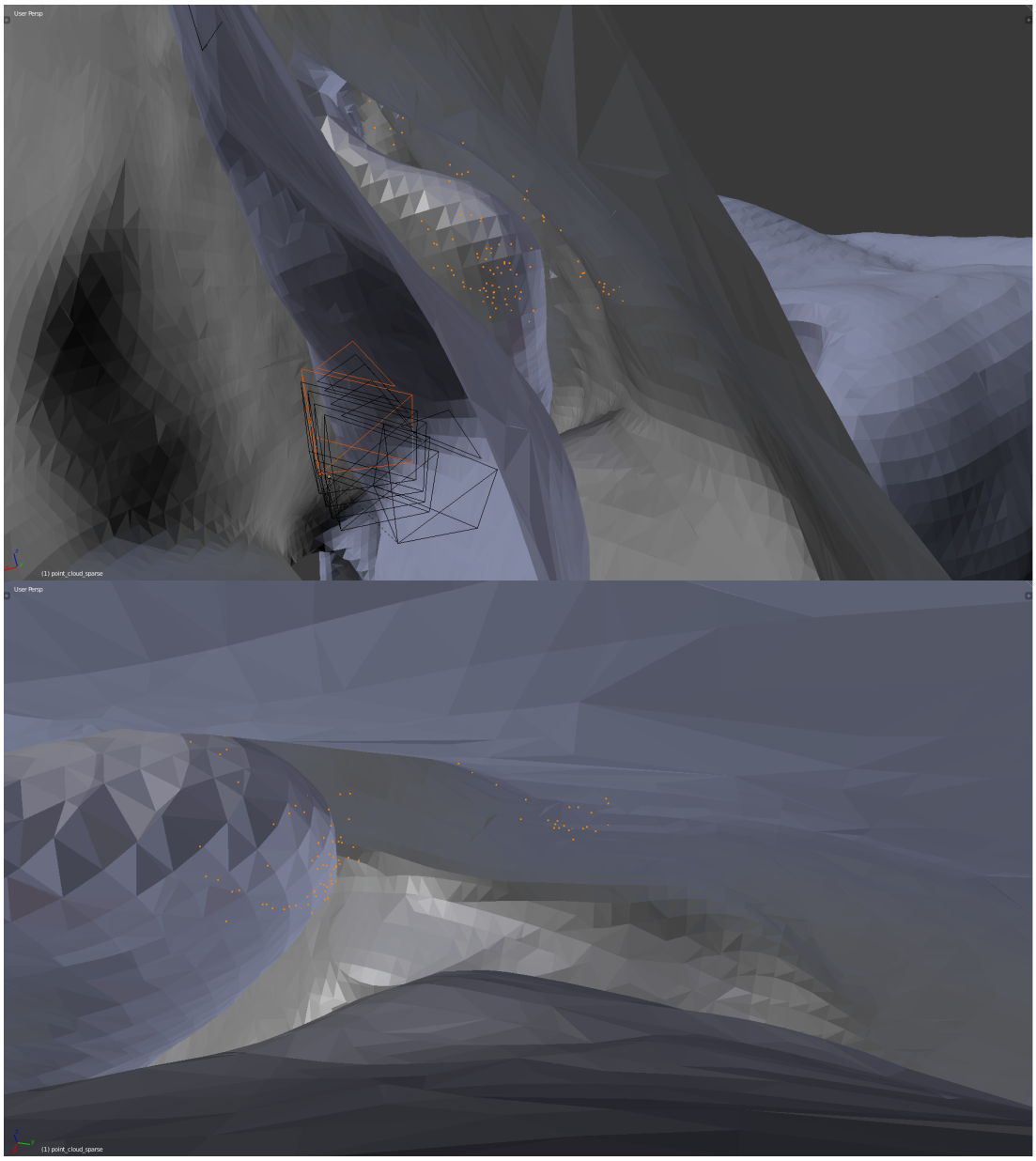

Figure 8.6: Experiment 1: the final alignment of the clinical video data, as registered by the Trimmed ICP algorithm, showing the locations of the 3D SFM feature points and of the camera poses. The upper portion of the figure shows a view looking down the airway into the sinus from a vantage point near the left nostril. The mesh in view in the upper figure is cropped forward of the vantage point in order to see both within the sinus airway as well as outside the medial (left) and lateral (right) walls of the sinus airway. The lower portion of the figure shows a view from above and looking towards the middle turbinate, similar to the upper portion of the initial alignment shown in Figure 8.5. Shown in highlight are the front camera and the 3D SFM feature points. Note that the registered camera positions have left the feasible region of the sinus airway.

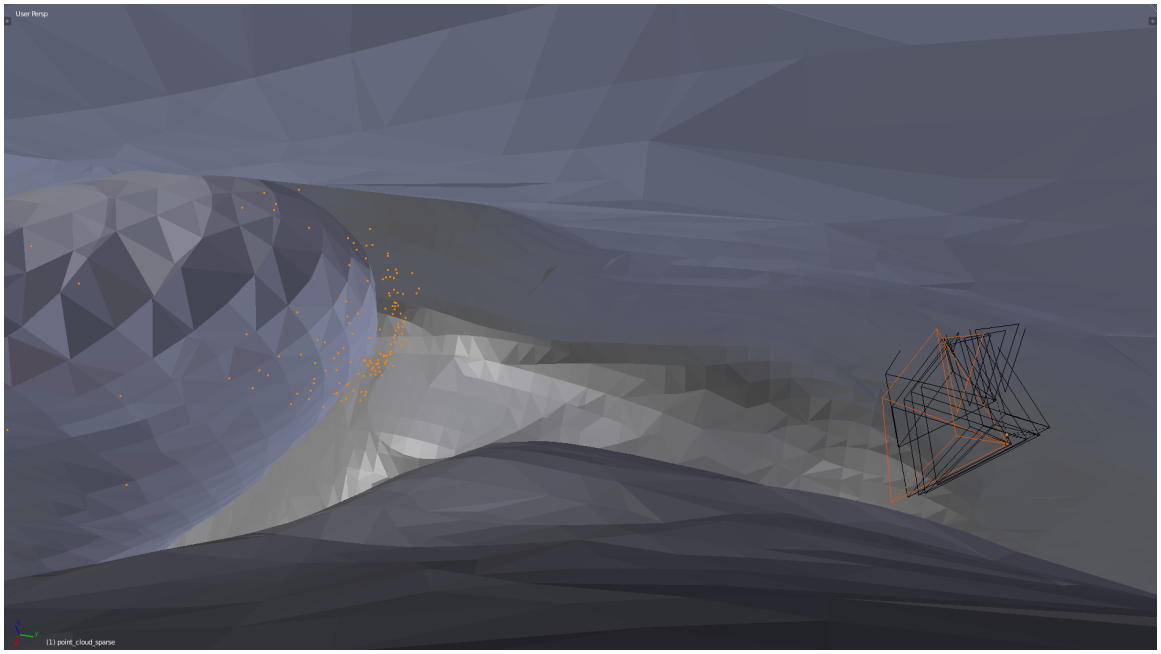

Figure 8.7: Experiment 1: the final alignment of the clinical video data, as registered by the V-IMLOP algorithm, showing the locations of the 3D SFM feature points and of the camera poses. The view is shown from above and looking towards the middle turbinate, similar to the upper portion of the initial alignment shown in Figure 8.5. Shown in highlight are the front camera and the 3D SFM feature points. Note that the registered camera positions have remained within the feasible region of the sinus airway.

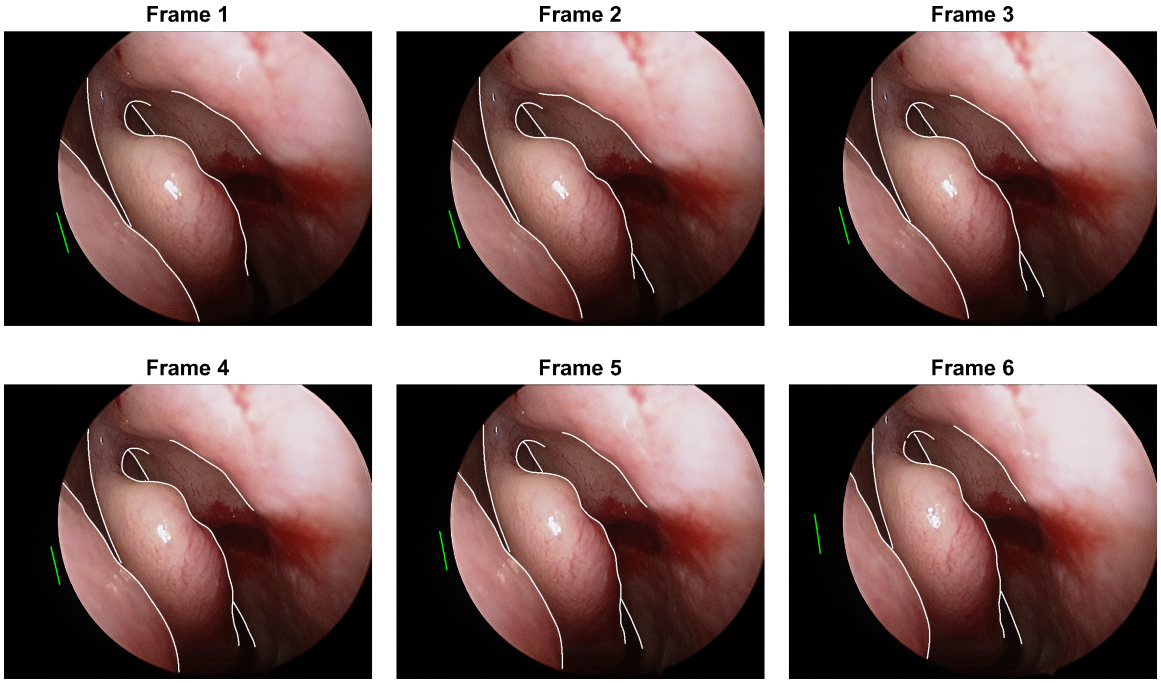

Figure 8.8: Experiment 1: the final alignment of the clinical video data, as registered by the Trimmed ICP algorithm, showing the estimated model-shape contours (as they were computed at the final estimated camera poses) projected onto the video images. The projected model-shape contours are shown in green overlay. The manually segmented video contours are also shown in white overlay. Very few projected contours exist, due to the registered camera positions being located outside of the sinus airway. Note that these video contours are not used within the Trimmed ICP algorithm; they are shown here only in the interest of investigating the registered alignments of the final camera poses.

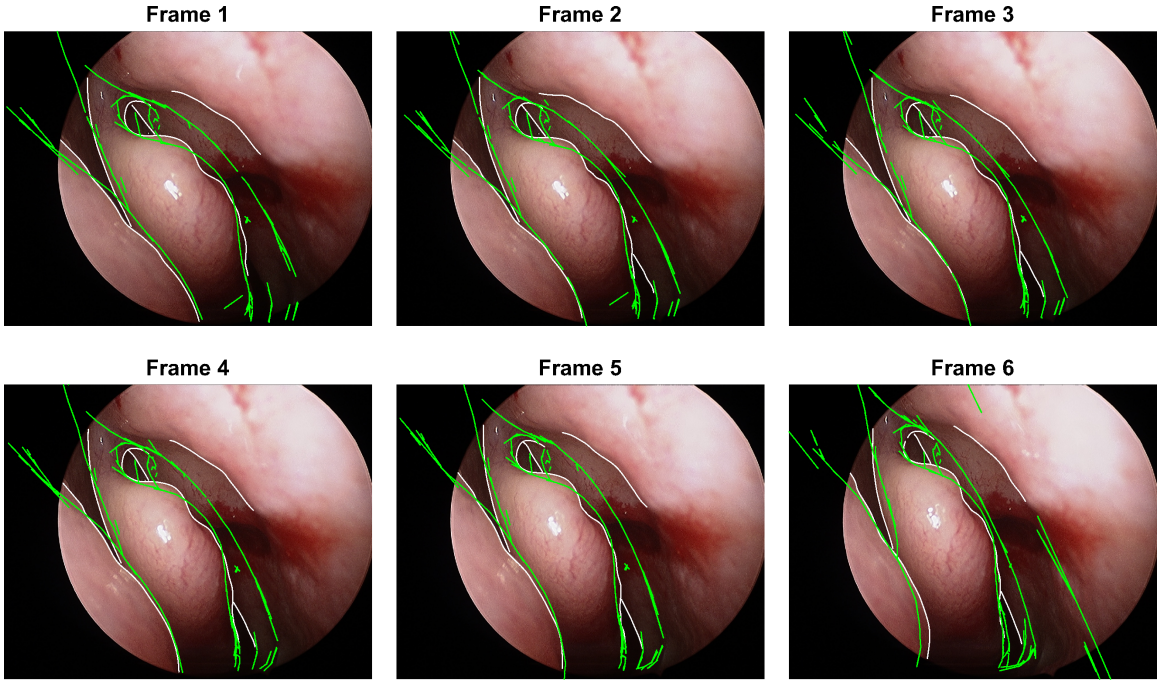

Figure 8.9: Experiment 1: the final alignment of the clinical video data, as registered by the V-IMLOP algorithm, showing the estimated model-shape contours (as they were computed at the final estimated camera poses) projected onto the video images. The projected model-shape contours are shown in green overlay. The manually segmented video contours are also shown in white overlay. These overlays represent the two sets of contours that were actively registered by the V-IMLOP algorithm.

points are clustered near the surface of the middle turbinate and that the camera positions were initialized at feasible locations within the sinus airway. In addition to the grouping of SFM points near the middle turbinate, additional SFM points were located outside the lateral wall of the sinus, which do not appear in the view shown by this figure; these points begin to appear in the registered alignment shown in Figure 8.6.

Concerning the registration computed by the Trimmed ICP algorithm, Figure 8.6 clearly shows that although the 3D SFM feature points appear to be accurately registered, the estimated camera positions ended up at infeasible locations outside the medial sinus wall. Because the estimated camera positions exited the sinus airway with the cameras facing back towards the medial wall, very few visible model-shape contours project to these camera poses, as seen in Figure 8.8.

Concerning the registration computed by the V-IMLOP algorithm, Figure 8.6 shows that the camera positions remained feasible (which is not surprising given the anatomical constraint). It is interesting to note (concerning this particular registration) that the anatomical constraint was not encountered during the registration, however, and the V-IMLOP algorithm would have computed the same outcome without it. By appearance, the 3D SFM points appears to be in worse alignment than the Trimmed ICP outcome, even though the Trimmed ICP solution is infeasible, noting that the V-IMLOP algorithm registered the 3D SFM points slightly above the surface of the middle turbinate. The projected model-shape contours shown in Fig-

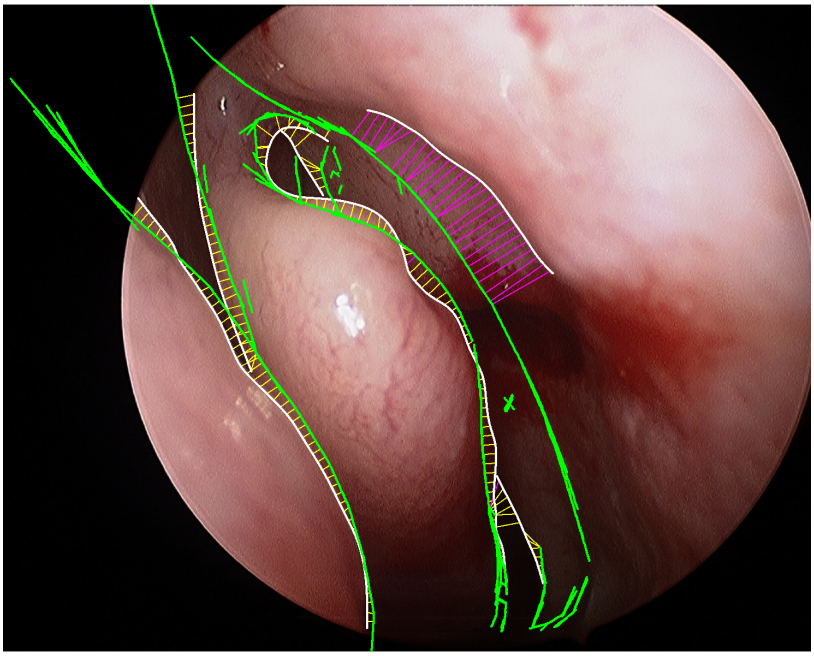

Figure 8.10: Video frame 5 from Experiment 1 showing the final matches computed by the V-IMLOP algorithm between the segmented video contours (white overlay) and the estimated model-shape contours (green overlay). The yellow lines connect matches that the V-IMLOP algorithm considered to be inliers, whereas the magenta lines connect matches that the V-IMLOP algorithm considered to be outliers.

ure 8.9, appear reasonably well aligned with the contours in the video images, with a discrepancy concerning one of the contours segmented from the lateral (right side of the image) wall. This contour, whose overlay is shown in the upper central region of each image in Figure 8.9, was disregarded by the V-IMLOP algorithm as an outlier. Figure 8.10 shows this in close-up detail for video frame 5, where the final matches between the video contours and the estimated contours from the model are shown by yellow connective lines for the set of inliers and by magenta connective lines for the set of outliers. The spacing between the these lines also illustrates the sampling density of the segmented video contours.

The residual misalignment seen in the registration from the V-IMLOP algorithm may be due to a complication regarding the registered patient data, which arises from congestion of the sinus tissue. Prior to endonasal surgery, patients undergo a decongestion procedure in order to fully dilate the sinus airway. The decongestion procedure is important because the tissues of the human sinus naturally follow cycles of congestion and decongestion throughout the day. Tissues constituting the regions of the interior sinus walls within the video images used in this study, as well as the middle turbinate, are affected by congestion, which alters the physical shape of the tissue. The patient represented in this study was decongested prior to the video images being acquired. However, no decongestion was administered prior to acquisition of the preoperative CT image, which would have been acquired days in advance of the operation. Thus, it is expected that the contours seen in the video will not have perfect alignment with the estimated contours from the 3D model-shape, which was generated from the preoperative CT image. This same concern also applies to the alignment of the 3D SFM feature points.

8.6.2 Experiment 2: Registering Patient Data with Simulated Video Features

The prior experiment suffered from the drawback of having an unknown ground truth, as well as complications in the data due to congestion of the sinus tissues. To

address this shortcoming, this experiment evaluates the performance of the V-IMLOP algorithm under known ground truth and without congestion-related complications by simulating the video-based features to be registered from the surface model of the patient's sinus that is generated from the preoperative CT. As before, this surface model also serves as the model shape for the registration.

The experiments is performed by simulating the positions of virtual cameras at known poses with respect to the sinus surface model. For each pose, a set of 3D feature points are then randomly sampled from the region of the surface mesh that is visible to the virtual camera in order to simulate the ground-truth 3D feature positions computed via SFM. The mesh regions that are visible to each virtual camera are obtained by using the DirectX framework to first render and then crop the visible parts of the mesh to the virtual camera's field of view, in similar manner as described in [127]. Noise is then added to these points to simulate noisy 3D SFM feature data. In order to simulate contours that have been segmented from the video images, the DirectX framework is also used to project the occluding contours from the mesh onto the virtual image planes; these initial contours are then used to simulate the segmented video contours. After computing the simulated video contours, noise is added to the ground truth camera poses in order to simulate noise in the extrinsic parameters that would have been present if the camera poses had been estimated via SFM. The noise assumed for the camera poses is much lower than the noise assumed for the SFM features, since in reality each camera pose would be estimated from many

SFM features and should therefore have less error. Registrations are conducted by randomly misaligning the simulated data from the sinus mesh and then registering the data back to the mesh in an attempt to recover the ground-truth alignment.

This experiment simulates the scenario of the prior study by using six virtual cameras positioned in view of the middle turbinate of the patient's left sinus. For the simulated 3D SFM features, 900 points were generated from the mesh. For half of the SFM feature points, noise was generated with 0.5 mm standard deviation in the virtual camera view direction and 0.3 mm standard deviation in the image parallel directions. For the other half of the SFM feature points, noise was generated by offsetting the sampled points along their associated surface normal directions by a magnitude uniformly distributed on the interval [-2.5, 2.5] mm. This sampling technique provides a large number of feature points close to the mesh surface along with a significant number of outliers positioned further from the mesh surface as may be expected from real SFM data. Since it is more difficult to predict the level of noise that may be associated with the extrinsic camera parameters, several noise levels are tested; misalignment magnitudes for the extrinsic parameters were uniformly generated on the intervals $[0, 0]$, $[0, 0.05]$, $[0.05, 0.1]$, $[0.1, 0.15]$, and $[0.15, 0.25]$ degrees (mm) and applied in random directions for generating the rotation (translation) errors. Different pose errors were independently generated for each virtual camera. The pose errors were applied to the cameras such that the center of rotation was positioned at each camera's optical center. After applying the individual pose errors

to each camera, a random global misalignment was generated to simulate an initial misalignment of the data from the sinus mesh, having an offset magnitude generated uniformly from the interval $[2, 3]$ degrees (mm) of rotational (translational) error. All misalignments were computed such that the virtual camera positions remained within the feasible region of the sinus airway. If a random misalignment violated this condition, the misalignment was simply regenerated until a feasible initial misalignment was found. No scaling error was added to the data.

The termination conditions for both algorithms were set as described for the prior study, except that the maximum iteration counts were set to 150 and 200 for the V-IMLOP and Trimmed ICP algorithms, respectively. Other algorithm settings remained as described for the prior study, except that constraints on scale estimation were enabled for the V-IMLOP algorithm using the constraint interval $[0.9, 1.1]$, which allows for deviations of $\pm 10\%$ from the initial scale of unity.

Thirty randomized trials were conducted for each range of camera pose error. Within each trial, a new set of SFM features, camera pose errors, and global misalignment were generated. The misaligned data was then registered by both the V-IMLOP and Trimmed ICP algorithms. The data used for the Trimmed ICP algorithm was the same data generated for the V-IMLOP algorithm at the lowest camera-pose-error interval. Different data was generated for running the V-IMLOP algorithm at each of the higher camera-pose-error intervals. The registration accuracy was assessed by taking points from the mesh as validation points to compute an average TRE. The

set of validation points was formed from the center points of every mesh triangle that was visible to one of the virtual cameras at the ground-truth pose.

Figure 8.11 shows a histogram of the average TREs computed by each algorithm. For the V-IMLOP algorithm, five histograms are shown corresponding to each camera-pose-error interval. Only one histogram is shown for the Trimmed ICP algorithm because this algorithm is not affected by the camera pose errors, which in this study only affect the alignment of the video contours, since the noise for the SFM features is generated independently. As seen in Figure 8.11, in all cases the V-IMLOP algorithm has significantly improved accuracy compared to the Trimmed ICP method. The TREs for the V-IMLOP algorithm increase steadily with increasing camera pose error. The registration accuracy of the V-IMLOP algorithm starts out at approximately 0.2 mm for the case of zero camera pose error and increases to approximately 1 mm for the highest magnitude of camera pose error ([0.15, 0.25] degrees (mm) of rotation (translation) error). The registration accuracy of Trimmed ICP, on the other hand, is for the most part in the range of 2–3 mm.

8.7 Concluding Remarks

The work of this chapter has focused on developing the V-IMLOP registration procedure and completing proof-of-concept experiments for the algorithm. While further experiments are required to fully assess the performance of this algorithm, the exper-

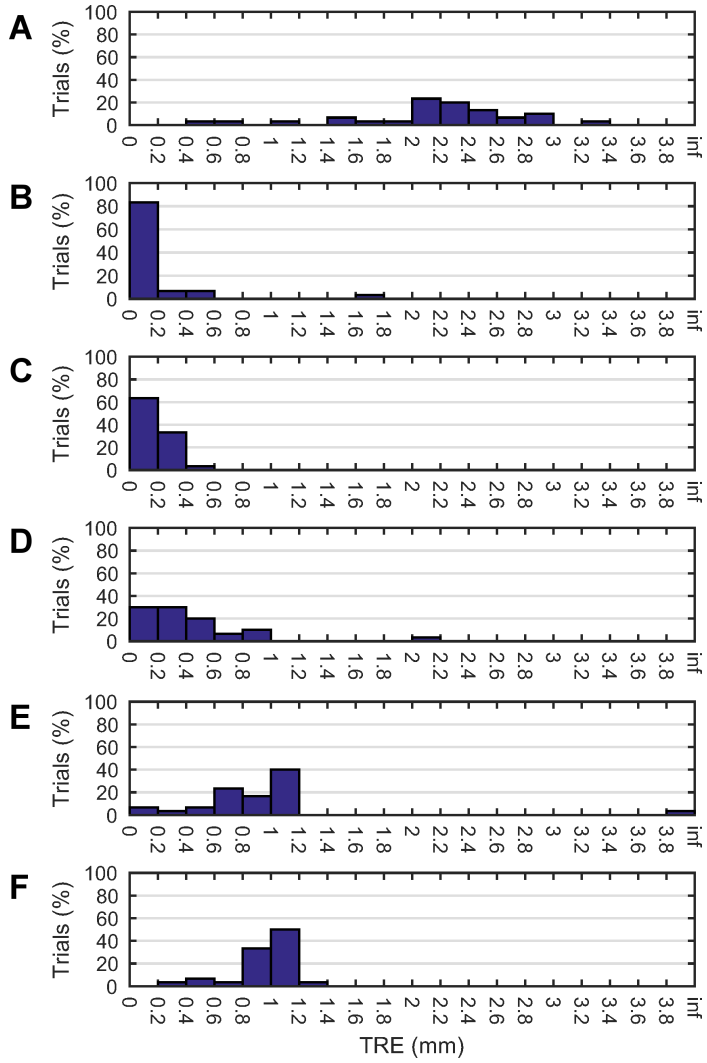

Figure 8.11: Experiment 2: histograms of the TRE outcomes from the registration trials using simulated video features for registration. Each histogram shows the average TRE values from 30 randomized trials corresponding to the registration outcomes for the (A) Trimmed ICP algorithm and for the V-IMLOP algorithm with camera pose error magnitudes drawn from the intervals of (B) [0, 0], (C) [0, 0.05], (D) [0.05, 0.1], (E) [0.1, 0.15], and (F) [0.15, 0.25] degrees (mm) of rotational (translational) offset error, respectively.

iments from this chapter using real and simulated patient data have demonstrated strong potential for the V-IMLOP algorithm to improve the accuracy of video-based registration for endoscopic procedures.

There are many areas for future work. In particular, since the focus of this work was to develop the registration algorithm rather than compute the features to be registered, it remains to investigate techniques for automatically segmenting contours from the video images. In the experiments presented in this chapter, these contours were either manually segmented from the video images or generated by simulation.

Techniques could also be investigated to account for changes in tissue congestion between the video and preoperative CT data. The most simple solution is to decongest the patient prior to performing the preoperative CT scan. However, this may not always be practical. Therefore, it is desirable to be able to correct for this condition. One possible solution is to model the change in the congested/decongested tissues using a statistical shape model. The surface mesh used in the registration could then be deformed according to the modes of the statistical shape model in order to adjust for different states of congestion. This deformation could be applied statically prior to starting the registration, or applied dynamically by directly optimizing the statistical parameters of the mesh within the registration procedure (which is described in the next chapter).

Another area for future work is to more aptly model the uncertainty in the SFM and contour features. In this study, the SFM features were characterized by a simple

noise model that assumed higher uncertainty in the camera depth direction. Ideally, each SFM feature point would have its own completely independent uncertainty, which could account for such factors as the strength of the underlying SIFT feature matches and the amount of reconstruction error arising from the SFM procedure, the depth of the reconstructed SFM point, etc. The uncertainty models for both the SFM and contour features could also consider whether the feature represents tissue that is affected by congestion and the degree by which that region of tissue is affected.

As a final point of discussion, in the derivation of the V-IMLOP algorithm presented in this chapter, the uncertainty of the 3D SFM features was set to a constant value relative to the dimensions of the model. If estimating the uncertainty of the SFM features based on the residual errors of the SFM procedure, then it may be more appropriate to scale the uncertainty of the SFM features along with the scaling of the data shape.

8.8 Contributions

The contributions from this chapter include:

6. Video IMLOP (V-IMLOP) Algorithm

   (a) a robust probabilistic algorithm for registering *video-image-based* features to a 3D surface model, while supporting anatomical constraints on the registration solution; this algorithm registers 3D *scaled position* features and

2D *perspective projected* contour features that consist of *position* and *orientation* components measured from video images; *anisotropic* uncertainty is supported for the positional features

(b) an efficient implementation of the V-IMLOP algorithm consisting of:

i. V-IMLOP Correspondence Phase

A. *Estimating the Occluding Contours of the Model Shape under Perspective Projection*: a GPU-based software framework for computing the perspective-projected occluding contours of a model shape as viewed from the imaging planes of a given set of virtual camera poses

B. *Computing the Most Likely Matches on the Model Shape*: fast PD-tree-based search techniques for efficiently computing the most likely matches from the model shape for the SFM features and for efficiently computing the most likely matches from the estimated contours of the model shape for the video contour features, while accounting for anisotropic data uncertainties and while accounting for both the position and orientation components of the contour features

ii. V-IMLOP Registration Phase with Anatomical Constraints

A. *Computing the Optimal Unconstrained Similarity Transformation to Align the Matched Feature Sets*: a gradient-based solution for

the problem of registering the SFM and contour features with the corresponding features from the model shape while accounting for anisotropic data uncertainties and while accounting for both the position and orientation components of the contour features; the solution is formed by computing the gradient equations of the objective function, which is optimized using an off-the-shelf, nonlinear, quasi-Newton optimizer

B. *Enforcing Anatomical Constraints on the Registration Solution*: incorporation of anatomical constraints on the computed transformations in order to restrict the estimated camera positions to feasible regions of the anatomy

Chapter 9

Deformable Most Likely Point Registration

This chapter describes a general approach for integrating deformable registration within the probabilistic frameworks of the registration algorithms described in the prior chapters. This chapter focuses on the concepts for applying the deformable registration extension rather than focusing on the implementation details for every algorithm. Where helpful, a worked-out example is provided for the IMLP algorithm in order to illustrate the concepts of the deformable extension in concrete form.

Deformable registration is a vast and highly active research topic, with a long history of prior art. An overview of deformable methods and techniques has been presented by Crum et al.[1] Deformable registration has many clinical applications. An example application has already been suggested in the concluding remarks of

Chapter 8 for deforming a surface model of a patient's sinus in order to account for tissue variation due to congestion. Other examples include registering medical images from different patients, where certain anatomical differences are sure to exist or registering images taken of the same patient in order to monitor changes in the shapes of organs and tissues over time. Deformable registration concerns are also common to intraoperative navigation regarding soft tissue interventions, such as within the abdominal region where insufflation of the abdomen, patient breathing activity, and also surgery may cause changes in the size and shapes of the organs and tissues.

As described in the Introduction (Chapter 1), deformable methods vary significantly depending on the type of transformation employed to deform the shape of interest. The simplest form of deformation is the *similarity transformation*, which applies a global scaling to the shape. This simple form of deformation was incorporated in the V-IMLOP algorithm of Chapter 8 for registering video-based features known only to scale. In this chapter, we investigate a more complex form of deformation that allows local regions of the shape to vary in different ways. Local methods of shape deformation typically incorporate a mechanism to restrict the deformations to be locally smooth, such as by embedding a regularization term in the registration cost function. In this chapter, we investigate deformations based on statistical shape models (SSMs), which implicitly restrict deformation smoothness based on the modes of shape variation. SSMs are a well-studied and understood topic. An in-depth treatment has been provided by Davies et al.,[131] and many of the terms and definitions

used in this chapter follow from their approach.

The basic idea behind using a SSM to describe shape deformation is to treat the shape as a mean shape that can be deformed by adding shape-change vectors, which are defined by statistical modes of variation. In order to understand how this plays out, it is instructive to consider how an SSM is created. We consider the standard case of an SSM formed by Principal Component Analysis (PCA). Suppose that a set of shapes are had which represent various deformations of the same object, and suppose that an SSM is desired in order to model the range of deformations exhibited among this set of shapes. Further suppose that every shape has the same topology (i.e., the same mesh structure) and that the correspondences between each shape in the set are known (i.e., if given any point on one of the shapes then the homologous point on every other shape can be readily determined). For SSM construction, we assume that each shape is fully described by a set of points; for a mesh this is the set of triangle vertices, which for a given mesh are denoted by $\{\mathbf{v}_i\}$. The complete set of vertices forming a mesh are also denoted by the following notation

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{n_\mathrm{v}} \end{bmatrix} \tag{9.1}$$

where $\mathbf{V}$ represents the ordered, stacked vector of all vertices in the mesh. The SSM is constructed by first computing the mean shape, which is comprised of the mean

positions of each vertex. Thus, the mean shape is defined by

$$\bar{\mathbf{V}} = \frac{1}{n_{\mathrm{s}}} \sum_{j=1}^{n_{\mathrm{s}}} \mathbf{V}_j \tag{9.2}$$

where $\mathbf{V}_j$ denotes the stacked vector of vertices for the jth mesh in the set of shapes. The next step is to compute the covariance of the vertex positions, which is referred to as the *shape covariance matrix* and is denoted by

$$\mathbf{\Sigma}_{\mathrm{SSM}} = \frac{1}{n_{\mathrm{s}}} \sum_{j=1}^{n_{\mathrm{s}}} (\mathbf{V}_j - \bar{\mathbf{V}})^{\mathsf{T}} (\mathbf{V}_j - \bar{\mathbf{V}}) \ . \tag{9.3}$$

Performing an eigen decomposition on the shape covariance matrix, $\mathbf{\Sigma}_{\mathrm{SSM}}$, provides the modes of shape variation

$$\mathbf{\Sigma}_{\mathrm{SSM}} = \begin{bmatrix} \mathbf{m}_1 & \dots & \mathbf{m}_{n_{\mathrm{s}}} \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_{n_{\mathrm{s}}} \end{bmatrix} \begin{bmatrix} \mathbf{m}_1 & \dots & \mathbf{m}_{n_{\mathrm{s}}} \end{bmatrix}^{\mathsf{T}} \tag{9.4}$$

where the orthonormal set of eigenvectors, $\{\mathbf{m}_i\}$, represent the *modes of variation*, which are also referred to simply as the *modes* and where the eigenvalues, $\{\lambda_i\}$, are the *mode weights*, which represent the amount of shape deformation that exists along the (n_{s}-dimensional) direction of each mode. Thus, the primary modes that explain the greatest amount of shape deformation are those that correspond to the largest mode weights. Note that the mode weights, $\{\lambda_i\}$, are listed by order of decreasing

magnitude. Equation (9.4) assumes that the number of shapes (n_{s}) are fewer than the number of vertices (n_{v}) comprising each mesh. Together, the mean shape, the modes, and the mode weights fully define an SSM model.

Using the mean shape and the modes of variation from the SSM model, any shape from the original set of shapes used to create the SSM model can be fully reconstructed using the following formula

$$\mathbf{V} = \bar{\mathbf{V}} + \sum_{i=1}^{n_{\mathrm{s}}-1} b_i \mathbf{m}_i \tag{9.5}$$

where $\{b_i\}$ are the *shape parameters* that define the components of the variation from the mean shape that lie along each of the modes directions, $\{\mathbf{m}_i\}$. Therefore, the shape parameters are defined by

$$b_i = \mathbf{m}_i^{\mathsf{T}}(\mathbf{V} - \bar{\mathbf{V}}) \ . \tag{9.6}$$

Note concerning Equation (9.5) that the last mode is not included when reconstructing the original shape from the SSM; this is because the final mode weight, $\lambda_{n_{\mathrm{m}}}$, is necessarily zero due to the mean shape and the shape covariance matrix being computed from the same set of shapes. By summing over fewer modes in Equation (9.5), an approximation to the original shape can also be obtained from the SSM, thereby reducing the amount of overhead for computation and data storage.

In the discussions that follow, the following alternative form for Equation (9.5) is

used

$$\mathbf{V} = \bar{\mathbf{V}} + \sum_{i=1}^{n_{\mathrm{m}}} s_i \mathbf{w}_i \tag{9.7}$$

$$\mathbf{w}_i = \sqrt{\lambda_i}\mathbf{m}_i \tag{9.8}$$

$$s_i = \mathbf{w}_i^{\mathsf{T}}(\mathbf{V} - \bar{\mathbf{V}}) \tag{9.9}$$

where $\{\mathbf{w}_i\}$ are the *weighted modes of variation* and where $\{s_i\}$ are the shape parameters in units of standard deviation relative to the SSM covariance. Note that in this formulation the number of modes is an arbitrary value chosen by the user, which may be any positive natural number less than the number of shapes used to generate the SSM (n_{s}), assuming there are fewer shapes than vertices representing each shape. The following definition for the deformable transformation operator ($\mathrm{T}_{\mathrm{ssm}}(\cdot)$) applied to a mesh vertex ($\bar{\mathbf{v}}_j$) taken from the mean SSM shape will be referenced in the following discussions

$$\mathrm{T}_{\mathrm{ssm}}(\bar{\mathbf{v}}_j) = \mathbf{v}_j = \bar{\mathbf{v}}_j + \sum_{i=1}^{n_{\mathrm{m}}} s_i \mathbf{w}_i^{(j)} \tag{9.10}$$

$$\mathbf{w}_i = \begin{bmatrix} \mathbf{w}_i^{(1)} \\ \vdots \\ \mathbf{w}_i^{(n_{\mathrm{v}})} \end{bmatrix} \tag{9.11}$$

where $\mathbf{w}_i^{(j)}$ is the component of the weighted mode, $\mathbf{w}_i$, that corresponds to the jth vertex, $\mathbf{v}_j$.

9.1 Probabilistic Model for Shape Deformation

In this section, an expression is derived for assuming a probability on an instance of a model shape that has been generated from a SSM. Assigning a probability to the model shape enables the shape deformation parameters to be directly optimized within the probabilistic frameworks of any of the prior registration algorithms presented in this dissertation.

The shape probability is formed by assuming a Gaussian distribution on the deformation from the mean shape. Given this assumption, the likelihood of a deformed vertex is defined by

$$f_{\text{vertex}}(\mathbf{v}\,;\mathbf{s}) = \prod_{i=1}^{n_{\text{m}}} \frac{1}{(2\pi)^{3/2}} \cdot e^{-\frac{\|s_i\|_2^2}{2}} \tag{9.12}$$

where $\mathbf{s}$ represents the set of weighted shape parameters, $\{s_i\}$, for the shape to which the vertex, $\mathbf{v}$, belongs. Note that because the weighted shape parameters, $\{s_i\}$, are already in units of standard deviation, no variance parameter is required for the Gaussian distribution of Equation (9.12) (i.e., the variance is simply one). Although the vertex position, $\mathbf{v}$, is not required in order to compute the vertex probability (i.e., only the shape parameters, $\mathbf{s}$, are required), the vertex is nonetheless included in the expression, $f_{\text{vertex}}(\mathbf{v}\,;\mathbf{s})$, in order to signify that the expression represents the probability of the vertex "given" parameters from an SSM-based deformation model for

that vertex. As will be discussed later in this chapter, this expression for the probability of a vertex could be adapted to other non-SSM-based shape deformations, in which case the shape parameters, $\mathbf{s}$, would be changed to the appropriate parameters for that deformation. If the assumption is made that the deformation of each vertex representing a shape is independent, then it follows that the likelihood of a complete shape, which is represented as a set of vertices, $\mathbf{V}$, is defined by the following *shape likelihood function*

$$f_{\text{shape}}(\mathbf{V}\,;\mathbf{s}) = \prod_{i=1}^{n_{\text{v}}} f_{\text{vertex}}(\mathbf{v}_i\,;\mathbf{s}) \tag{9.13}$$

In reality, this assumption is not completely accurate, as there is likely to be some local dependence between the deformations for vertices representing the same shape. However, this assumption enables a tractable formulation of the deformable-shape probability, and also becomes more valid as the sparsity of the vertices representing the shape increases. This assumption may also become more valid when used within the context of the full probabilistic model for deformable registration, as we will see later in the coming section.

9.2 Probabilistic Model for Deformable Registration

Given the deformable-shape probability model derived in Section 9.1, a probabilistic model that combines the shape likelihood with the match likelihoods of the registration algorithms can now be derived. Assuming independence between the data-shape matches and the deformation of the model shape leads to the following *deformable match likelihood function*

$$f_{\text{match\_deformable}}(\mathbf{x}, \mathbf{y}\,; \theta, \mathbf{s}, \bar{\mathbf{V}}, \mathbf{W}) = f_{\text{match}}(\mathbf{x}\,; \mathrm{T}_{\text{ssm}}(\mathbf{y}), \theta) \cdot f_{\text{shape}}(\mathrm{T}_{\text{ssm}}(\mathbf{y})\,; \mathbf{s}) \quad (9.14)$$

and the following *deformable total match likelihood function*

$$f_{\text{match\_deformable}}(\mathbf{X}, \mathbf{Y}\,; \theta, \mathbf{s}, \bar{\mathbf{V}}, \mathbf{W}) = \left(\prod_{i=1}^{n_{\text{data}}} f_{\text{match}}(\mathbf{x}_i\,; \mathrm{T}_{\text{ssm}}(\mathbf{y}_i), \theta_i) \right) \cdot f_{\text{shape}}(\mathrm{T}_{\text{ssm}}(\mathbf{Y})\,; \mathbf{s}) \quad (9.15)$$

where θ represents the distribution parameters of the match likelihood function, which vary by the algorithm, and where $\mathbf{s}$ represents the model-shape parameters that define an instance of the model shape from the SSM. The SSM is represented by the matrix of weighted modes, $\mathbf{W}$, and by the mean shape, $\bar{\mathbf{V}}$. The expressions $\mathrm{T}_{\text{ssm}}(\mathbf{y}_i)$ and $\mathrm{T}_{\text{ssm}}(\mathbf{Y})$ represent an SSM-based deformable transformation of the model-shape matches, $\mathbf{Y}$. For the sake of analytical simplicity, we define a match point, $\mathbf{y}_i$, to be

the point on the mean SSM shape, $\bar{\mathbf{V}}$, that is homologous to the actual point of correspondence. The actual point of correspondence, $\mathrm{T}_{\mathrm{ssm}}(\mathbf{y}_i)$, is on the deformed shape and is computed by adding the current model-shape deformation corresponding to the mean point, $\mathbf{y}_i$. The mathematical definition for $\mathrm{T}_{\mathrm{ssm}}(\mathbf{y}_i)$ is further discussed in a following paragraph.

Unlike Equation 9.13, the formulations of Equations (9.14) and (9.15) define the likelihood of the shape deformation based on the current set of matches ($\mathbf{Y}$) rather than based on all of the vertices ($\mathbf{V}$) that comprise the model shape. This revised formulation for the deformable-shape probability has the advantage of normalizing the influence of the model shape likelihood component compared to the influence of the match likelihood component of the features being registered. If all of the vertices of the model shape were used, then increasing the sampling density of the points that form the model shape would increase the influence of the shape likelihood component within the registration, which is not the desired outcome. A related discussion regarding the normalization of different components of a match likelihood function is also found in Section 8.6, where a normalization was applied to balance the influence of the different types of features used to register video-based data. If the data-shape features are more sparsely sampled than the model-shape representation (which is often the case) then the assumption of independence between the deformations of the points used to represent the model shape also becomes more valid, as was alluded to at the end of Section 9.1.

We now turn to the mathematical definition for a deformably transformed match point. In this discussion it is assumed that the model shape is represented by a triangular mesh. All points on a triangle reside within the convex hull of the triangle's vertices. Thus, as shown in Equation (9.16), any point, $\mathbf{y}$, that is located on the face of a triangle may be represented as a weighted sum of the triangle vertices

$$\begin{aligned} \mathbf{y} &= \sum_{j=1}^{3} \mu^{(j)} \mathbf{v}^{(j)} \\ \text{subject to:} \quad & \sum_{i=1}^{3} \mu^{(j)} = 1 \end{aligned} \tag{9.16}$$

where the vertex weights, $\{\mu^{(j)}\}$, must sum to one and where $\{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \mathbf{v}^{(3)}\}$ are the three triangle vertices. Assuming that equations of the form shown in Equation (9.16) are used to define the set of match points, $\{\mathbf{y}_i\}$, then the SSM-based deformable transformation of a match point, $\mathrm{T}_{\mathrm{ssm}}(\mathbf{y}_i)$, is defined as

$$\mathrm{T}_{\mathrm{ssm}}(\mathbf{y}_i) = \sum_{j=1}^{3} \mu_i^{(j)} \, \mathrm{T}_{\mathrm{ssm}}(\mathbf{v}_i^{(j)}) \tag{9.17}$$

where $\mathbf{v}_i^{(j)}$ is the jth vertex of the mesh triangle on which the ith model point, $\mathbf{y}_i$, is located and where $\{\mu_i^{(j)}\}$ are the corresponding vertex weights. The deformable transformation, $\mathrm{T}_{\mathrm{ssm}}(\mathbf{v}_i^{(j)})$, of a mesh vertex, $\mathbf{v}_i^{(j)}$, was previously defined in Equation (9.10). Note that the deformable transformation operator, $\mathrm{T}_{\mathrm{ssm}}(\cdot)$, has the simplified definition of Equation (9.10) only when applied to vertex points and has the

general definition of Equation (9.17) for any other point on a triangle face that is not a vertex.

While the foregoing discussion has assumed the model shape to be a mesh, other representations of model shapes are also readily supported. In the case of a point-cloud representation of the model shape, for example, then the match points, $\{\mathbf{y}_i\}$, are essentially treated as "vertex" points, and the deformable transformation operator for the match points simplifies to the transformation of vertex points defined by Equation (9.10).

9.3 Deformable Correspondence Phase

In this section, the implementation of the correspondence phase for deformable versions of the registration algorithms is discussed. The deformable implementation is, in fact, straightforward, and the PD-tree search techniques for computing the matches remain the same for each algorithm. What changes is the positions of the points that define the model shape during each iteration. Since the topology of the model shape is constant with respect to deformation, the PD tree does not need to be reconstructed following a change in the shape deformation, and conceptually all that is required is to update the positions of the points that represent the model shape within the PD tree, as well as the bounds of the bounding boxes that surround these points within each node. Note that it does not matter if the overlaps between the

bounding boxes increase as a result of this update.

The PD tree may be constructed using the SSM positions defined by an initial set of values for the model-shape parameters, $\mathbf{s}$, which would be provided by the user. If no initial shape parameters are specified for the model shape, then the mean shape of the SSM may be used to construct the PD tree. At the beginning of each correspondence phase, the positions of the points representing the model shape must be recomputed based on the current values of the model-shape parameters, $\mathbf{s}$. Since the points defining the model shape would have moved from their prior locations, both the positions of these points within the PD tree and the bounds of the oriented bounding boxes for each PD-tree node must be updated. As an optimization, only the extent of the bounds for each bounding box may be updated, leaving the orientations of the bounding boxes unchanged, since small deformations of the model shape do not drastically affect the bounding-box orientations. Alternatively, the orientations of the bounding boxes could be updated if the accumulated deformation over several iterations changes by more than some user-defined threshold.

9.4 Deformable Registration Phase

In this section, the implementation of the registration phase for deformable versions of the registration algorithms is discussed. Conceptually, the only change is that the registration phase must now maximize the deformable total match likeli-

hood function of Equation (9.15) with respect to both the data-shape transformation parameters and the deformable model-shape parameters, rather than maximizing the total match likelihood function defined by the algorithm with respect to only the data-shape transformation parameters. Maximizing the deformable total match likelihood of Equation (9.15) is equivalent to minimizing its negative log, which reduces to minimizing the following *deformable total match error function*

$$\mathrm{E}_{\mathrm{match\_deformable}}(\mathbf{X}, \mathbf{Y}\,; \theta, \mathbf{s}) = \sum_{i=1}^{n_{\mathrm{data}}} \mathrm{E}_{\mathrm{match}}(\mathrm{T}(\mathbf{x}_i)\,; \mathrm{T}_{\mathrm{ssm}}(\mathbf{y}_i), \theta_i, \mathbf{s}) + \frac{n_{\mathrm{data}}}{2} \sum_{i=1}^{n_{\mathrm{m}}} \|s_i\|_2^2 \tag{9.18}$$

where $\mathrm{E}_{\mathrm{match}}(\cdot)$ represents the negative log of the match likelihood function, which is defined in the chapters corresponding to each algorithm, $\mathbf{T}(\mathbf{x}_i)$ represents the standard transformation of the data shape, as defined in the algorithm chapters, and $\mathrm{T}_{\mathrm{ssm}}(\mathbf{y}_i)$ represents the SSM-based deformable transformation of the match point, $\mathbf{y}_i$, which is defined in Equation (9.17). While optimizing over the deformable model-shape parameters, $\mathbf{s}$, it may also be desirable to constrain these values to a realistic range, such as ± 3 standard deviations from the mean shape for each mode.

9.4.1 Deformable Registration Phase Illustrated for the IMLP Algorithm

To grant a concrete illustration to the concept for the deformable registration phase, the optimization of the deformable total match likelihood function for the IMLP algorithm is now derived.

Substituting the IMLP match error expression from Equation (4.4) into the deformable total match error function of Equation (9.18) produces the following equation

$$\mathbf{T} = \underset{[\mathbf{R},\mathbf{t}],\mathbf{s}}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n_{\text{data}}} (\mathrm{T}_{\text{ssm}}(\mathbf{y}_i) - \mathbf{R}\mathbf{x}_i - \mathbf{t})^{\mathsf{T}} (\mathbf{R}\boldsymbol{\Sigma}_{\text{x}i}\mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\text{y}i})^{-1} (\mathrm{T}_{\text{ssm}}(\mathbf{y}_i) - \mathbf{R}\mathbf{x}_i - \mathbf{t}) + \frac{n_{\text{data}}}{2} \sum_{i=1}^{n_{\text{m}}} \|s_i\|_2^2 \right) \quad (9.19)$$

where a factor of $1/2$ has been added to the IMLP match error component, since in Equation (4.4) this factor was removed via a process of simplification. This objective function may be optimized by computing the gradient with respect to the optimization parameters and applying a nonlinear quasi-Newton based optimizer as demonstrated in prior chapters. In order to draw focus to the new derivative expressions introduced by the model-shape deformations, it will be assumed that the model-shape covariances $\{\boldsymbol{\Sigma}_{\text{y}i}\}$ are all zero. Applying this assumption and simplifying the constant terms

results in the following simplified objective function

$$\mathbf{T} = \underset{[\boldsymbol{a},\mathbf{t}],\mathbf{s}}{\operatorname{argmin}} \left(\sum_{i=1}^{n_{\text{data}}} C_{\text{match}i} + C_{\text{shape}} \right) \tag{9.20}$$

$$C_{\text{match}i} = \mathbf{z}_i^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{x}i}^{-1} \mathbf{z}_i \tag{9.21}$$

$$\begin{aligned} \mathbf{z}_i &= \mathrm{R}(\boldsymbol{a})^{\mathsf{T}} (\mathrm{T}_{\text{ssm}}(\mathbf{y}_i) - \mathrm{R}(\boldsymbol{a})\mathbf{x}_i - \mathbf{t}) \\ &= \mathrm{R}(\boldsymbol{a})^{\mathsf{T}} (\mathrm{T}_{\text{ssm}}(\mathbf{y}_i) - \mathbf{t}) - \mathbf{x}_i \end{aligned} \tag{9.22}$$

$$C_{\text{shape}} = n_{\text{data}} \mathbf{s}^{\mathsf{T}} \mathbf{s} \tag{9.23}$$

where we have also incorporated the Rodrigues formulation, $\mathrm{R}(\boldsymbol{a})$ (Equation (6.7)), in place of the rotation matrix, $\mathbf{R}$, in preparation for forming the gradient equations to be used within the optimization.

The expressions for the gradient (denoted by $\nabla\mathbf{C}$) of the deformable cost function of Equation (9.20) with respect to the transformation parameters, $[\boldsymbol{a}, \mathbf{t}]$, and with respect to the deformable model-shape parameters, $\mathbf{s}$, appear below. The gradient is expressed as a stacked vector with the transformation parameters located at the top, followed by the vector of model-shape parameters. The notation $\mathbf{J}_{a,b}$ signifies

the Jacobian of an expression, a, with respect to some variable expression, b.

$$\nabla \mathbf{C} = \sum_{i=1}^{n_{\text{data}}} \nabla \mathbf{C}_{\text{match}i} + \nabla \mathbf{C}_{\text{shape}} \tag{9.24}$$

$$\nabla \mathbf{C}_{\text{match}i} = \left[\mathbf{J}_{C_{\text{match}i},\mathbf{z}_i} \mathbf{J}_{\mathbf{z}_i,\boldsymbol{a}}, \quad \mathbf{J}_{C_{\text{match}i},\mathbf{z}_i} \mathbf{J}_{\mathbf{z}_i,\mathbf{t}}, \quad \mathbf{J}_{C_{\text{match}i},\mathbf{z}_i} \mathbf{J}_{\mathbf{z}_i,\mathbf{s}} \right]^{\mathsf{T}} \tag{9.25}$$

$$\mathbf{J}_{C_{\text{match}i},\mathbf{z}_i} = 2\, \mathbf{z}_i^{\mathsf{T}} \boldsymbol{\Sigma}_{\text{x}i}^{-1} \tag{9.26}$$

$$\mathbf{J}_{\mathbf{z}_i,\boldsymbol{a}} = \left[\frac{\partial \,\text{R}(\boldsymbol{a})}{\partial a_x}^{\mathsf{T}} (\text{T}_{\text{ssm}}(\mathbf{y}_i) - \mathbf{t}), \quad \frac{\partial \,\text{R}(\boldsymbol{a})}{\partial a_y}^{\mathsf{T}} (\text{T}_{\text{ssm}}(\mathbf{y}_i) - \mathbf{t}), \quad \frac{\partial \,\text{R}(\boldsymbol{a})}{\partial a_z}^{\mathsf{T}} (\text{T}_{\text{ssm}}(\mathbf{y}_i) - \mathbf{t}) \right] \tag{9.27}$$

$$\mathbf{J}_{\mathbf{z}_i,\mathbf{t}} = -\,\text{R}(\boldsymbol{a})^{\mathsf{T}} \tag{9.28}$$

$$\mathbf{J}_{\mathbf{z}_i,\mathbf{s}} = \mathbf{J}_{\mathbf{z}_i,\text{T}_{\text{ssm}}(\mathbf{y}_i)} \mathbf{J}_{\text{T}_{\text{ssm}}(\mathbf{y}_i),\mathbf{s}} \tag{9.29}$$

$$\mathbf{J}_{\mathbf{z}_i,\text{T}_{\text{ssm}}(\mathbf{y}_i)} = \text{R}(\boldsymbol{a})^{\mathsf{T}} \tag{9.30}$$

$$\mathbf{J}_{\text{T}_{\text{ssm}}(\mathbf{y}_i),\mathbf{s}} = \sum_{j=1}^{3} \mu_i^{(j)} \mathbf{J}_{\text{T}_{\text{ssm}}(\mathbf{v}_i^{(j)}),\mathbf{s}} \tag{9.31}$$

$$\mathbf{J}_{\text{T}_{\text{ssm}}(\mathbf{v}_i^{(j)}),\mathbf{s}} = \begin{bmatrix} 0 & \cdots & \mathbf{w}_i^{(j)} & \cdots & 0 \end{bmatrix} \tag{9.32}$$

$$\nabla \mathbf{C}_{\text{shape}} = \left[0, \quad 0, \quad 2n_{\text{data}} \mathbf{s}^{\mathsf{T}} \right]^{\mathsf{T}} \tag{9.33}$$

In the foregoing equations, $\frac{\partial \,\text{R}(\boldsymbol{a})}{\partial a_x}$ signifies the 3×3 matrix of partial derivatives of $\text{R}(\boldsymbol{a})$ with respect to element α_x of $\boldsymbol{a} = [\alpha_x, \alpha_y, \alpha_z]^{\mathsf{T}}$, and so on for α_y and α_z. Concerning Equation (9.32), note that the Jacobian of a vertex position, $\text{T}_{\text{ssm}}(\mathbf{v}_i^{(j)})$, with respect to the shape parameters, $\mathbf{s}$, is formed by positioning the weighted mode component, $\mathbf{w}_i^{(j)}$, at the ith column of a sparse $3 \times n_{\text{m}}$ matrix that contains zeros otherwise. In practice, one does not actually form these sparse matrices, but rather operates on the relevant non-zero components.

9.5 Concluding Remarks

In this chapter, a general approach was introduced for incorporating an SSM-based deformable registration component within any of the probabilistic registration algorithms presented in this dissertation. Implementation details were provided for one example—the IMLP algorithm.

Beyond SSM-based deformation, it is also possible to incorporate other types of deformation models. The approach remains the same. All that is required is a method for assuming a likelihood on the shape being deformed based on the deformation parameters. Once this likelihood has been defined, any form of deformation model may be used to form the deformable-shape likelihood within the deformable match likelihood function of Equation (9.15).

It is further possible to incorporate deformable models on both the data- and model-shapes. Such a formulation could be useful, for example, if the deformations in the two shapes follow different statistical distributions that could be aptly modeled by using two independent SSMs. Alternatively, some applications may find utility in incorporating different types of deformation models for each shape, not being limited to only SSMs.

9.6 Contributions

The contributions from this chapter include:

7. A general probabilistic-based approach for incorporating deformable shape transformations within the registration frameworks of the probabilistic algorithms developed for this dissertation

   (a) detailed development of a deformable registration approach for registering a deformable model shape that is characterized by a statistical shape model (SSM), where the shape deformations computed by the registration are driven by the modes of the SSM

   (b) a worked-out example of applying the general SSM-based deformable registration approach, illustrating a deformable version of the IMLP algorithm

Chapter 10

An Extensible Software Architecture for Rapid Development of ICP-Based Registration Algorithms

This chapter describes a software architecture that was developed in the course of this dissertation in order to enable rapid implementation of the various ICP-based algorithms of the prior chapters by both minimizing and encapsulating the code changes required to implement each algorithm. These design goals have been realized by creating an apt software design pattern for this application that incorporates sound principles of software design, such as abstraction, encapsulation, loose-coupling,

and well-defined interfaces using objected-oriented software features such as inheritance, polymorphism, composition, and abstract classes. The software design pattern described in this chapter has enabled much of the functionality of each ICP-based algorithm to be implemented within a stable core code base that does not change from one algorithm to the next and that requires no modification in order to add new algorithms. This is the design goal known as the *Open-Closed Principle* (i.e., "open for extension" and "closed for modification").[132]

The purpose of this chapter is not to document the code that was developed in the course of this dissertation per se, but rather to describe the key software design patterns that enabled efficiently implementing these various algorithms. Thus, minor aspects of the software framework as described in this chapter may differ from the actual implementation, especially variable and class names, which are anyway subject to change following this writing. The goal of this chapter is therefore to convey the high-level architecture of the software framework rather than the implementation details, which are many and varied.

It is planned that the code for the registration algorithms that have been developed for this dissertation will be made public in the near future and distributed from the same site that hosts the code for the CISST C++ libraries[72] (https://github.com/jhu-cisst).

10.1 C++ Software Architecture

The registration software framework is primarily implemented in the C++ programming language. C++ was chosen because of its runtime efficiency, object oriented design features, extensive user base within the field, and strong support community. The software framework is closely integrated with the CISST C++ libraries,[72] particularly the CISST Vector and CISST Numerical libraries, which provide foundational support for linear algebra and many numerical computations.

The software architecture is divided into two major components: a generic framework component, which implements the core code that is required by all algorithms, and an algorithm-specific component, which implements the code that varies from one algorithm to the next. The goal of this architecture is to keep these two software components loosely coupled, such that changes in the implementation of an algorithm or additions of new algorithms do not translate to code changes being required within the generic framework component and vice versa.

The generic framework and the algorithm-specific software components are linked via an abstract interface. The abstract interface defines the methods that each algorithm must implement, and these methods are invoked from the generic framework through the defined interface. This abstract interface is the only means by which the generic framework component interacts with the algorithm-specific component of the software; thus, with the exception of the interface definition, the implementations of the two software components are decoupled.

The interface between the generic framework and the algorithm-specific software components is created using the abstract class feature of C++. The abstract class that defines this interface is sub-typed by classes within the algorithm-specific code via inheritance. The code for each type of algorithm is encapsulated within a unique algorithm class. Because the algorithm classes are inherited from the abstract class, the algorithm classes are guaranteed to implement the proper software interface. Each algorithm class is required by the compiler to provide implementations of the methods that are defined by the software interface, because these methods are defined to be pure virtual within the abstract class.

The generic framework interacts with the algorithm-specific code by providing an instance of an algorithm class as an input to the generic framework when calling one of the generic framework routines. The algorithm object is represented to the generic framework as only an instance of the abstract class that defines the software interface. Thus, the generic framework has no knowledge of the specific algorithm-class type; it only knows that the algorithm object is an instance of the abstract class that defines the software interface and that it can therefore invoke methods on the algorithm object that have been defined by the software interface. When an interface method is invoked from the generic framework on the algorithm object, through polymorphism the method invocation is automatically directed to the specific implementation that is defined within the algorithm class.

10.1.1 Algorithm-Specific Software Component

Two primary types of abstract interfaces are used by the software. The first type of interface is the *ICP interface*, which defines the key algorithmic methods that are required by the registration loop. The second type of interface is the *PD-tree interface*, which defines the key algorithmic methods that are required by the PD-tree search. In addition to containing the definitions of methods that make up an interface, the abstract class that implements the interface may also contain data members that are needed by all of the algorithm classes that are derived from the abstract class.

The key elements of the ICP interface are illustrated in Figure 10.1; the abstract class that defines the ICP interface within this illustration is entitled "AlgorithmICP". Several algorithm classes that sub-type this interface are also shown as examples. In addition to the interface methods that must be defined by each sub-class, the interface also includes data members for the data- and model-shape feature positions, since this type of data is used by all of the algorithms that have been implemented. Other data members that are only needed for a specific algorithm are defined at the algorithm class level, such as the data- and model-shape noise-model covariances of the IMLP algorithm class or the orientation concentration and position variance parameters of the IMLOP class, etc.

The key elements of the PD-tree interface are illustrated in Figure 10.2; the abstract class that defines the PD-tree interface within this illustration is entitled "AlgorithmPDTree". In addition to the PD-tree search methods that must be defined by

each algorithm class, the abstract class that defines the PD-tree interface also includes a pointer to the PD-tree that represents the model shape, which is needed in order to reference the geometric elements of the model shape within the PD-tree search methods that are implemented by the algorithm-specific classes. The interface method named "NodePossiblyCloser()" is the method that computes whether to search or skip each node by comparing the bounding properties of the node to the current best match error. The interface method named "ComputeBestMatchOnDatum()" is the method that is called in order to compute the lowest match error for a single datum when comparing the match errors for individual datums within a leaf node of the PD tree. In practice, the implementation of this method varies by the type of datum that is used to represent the model shape (i.e., whether the mode shape is a point cloud or a mesh). Thus, the algorithm classes that are derived from the PD-tree interface are themselves sub-typed by an additional level of inheritance according to the type of model shape; this final level of inheritance is not shown in Figure 10.2.

10.1.2 Generic Framework Software Component

The key elements of the generic framework code are the routines that run the registration loop and that manage the PD-tree-search procedure. In particular, the routine that is called by the user in order to launch a registration is defined within the generic framework. As mentioned earlier, the generic framework routines take an algorithm object as input. The algorithm objects provide specific implementations of

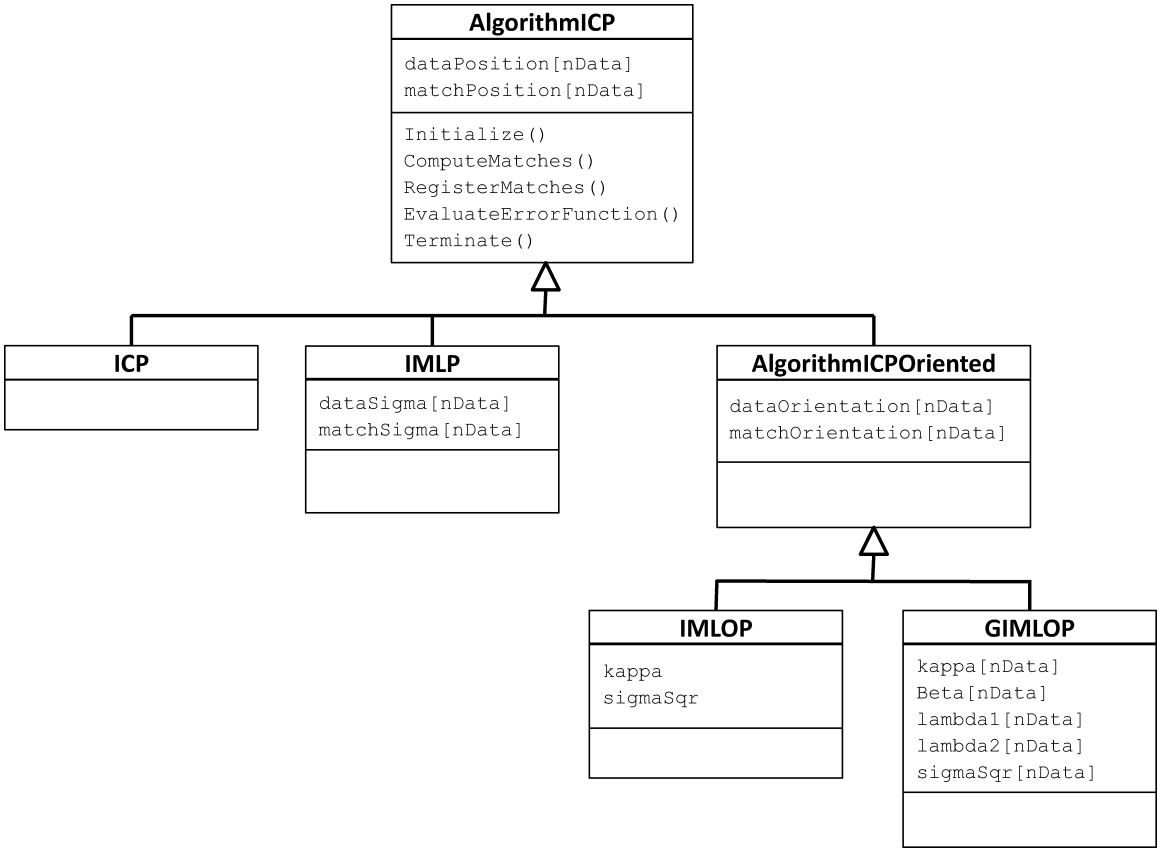

Figure 10.1: Illustration of the abstract "ICP interface" for creating classes that implement the algorithm-specific operations that are invoked during the ICP-based registration loop.

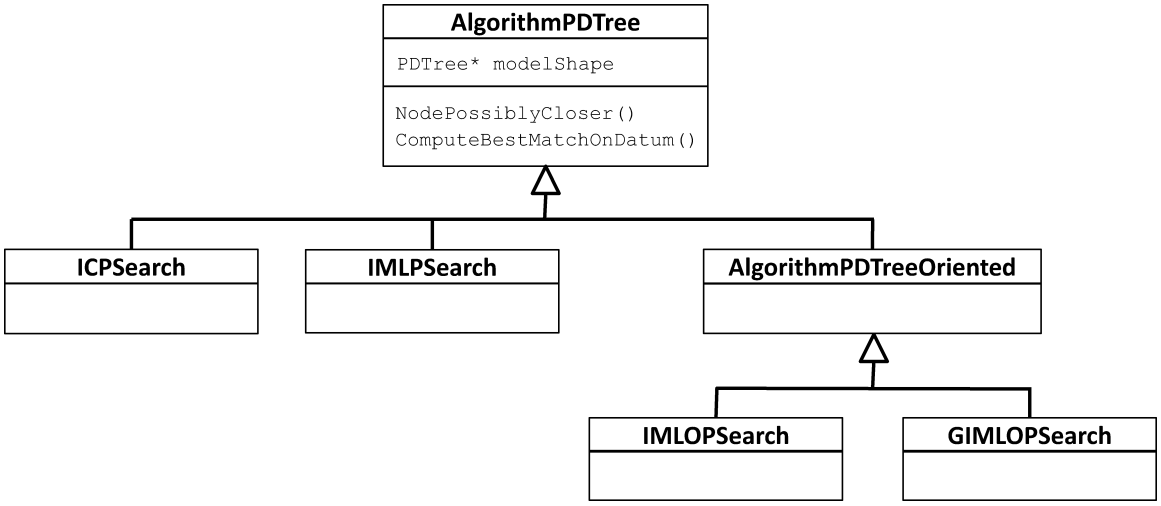

Figure 10.2: Illustration of the abstract "PD-tree interface" for creating classes that implement the algorithm-specific operations that are invoked during a PD-tree search.

the methods that control how the key algorithmic steps are performed, thereby enabling the behaviors of the generic procedures to vary from one algorithm to another.

The routines within the generic framework define the structure and sequence of the steps that are involved while performing an algorithmic procedure, such as the steps involved in iterating the registration loop of the ICP-based registration methods. Some of these steps remain the same regardless of the type of algorithm chosen, whereas other steps vary by the type of algorithm. The generic routines directly implement the steps that are invariable and encapsulate the steps that vary within method calls, which are invoked on the algorithm object that was provided as an input. This approach to structuring the software architecture was inspired by the strategy and template design patterns, which incorporate similar abstractions.[132]

The generic routine that is called by the user in order to run a registration is described by Algorithm 10.1. The routine begins by initializing the registration loop and the algorithm object. The initial guess for the registration transformation is sent to the algorithm object in order to initialize the alignment of the shape data, which is stored within the algorithm object. Then the registration loop is started, which iterates calls to the algorithm methods that implement the correspondence and registration phases. Support is given for an optional set of user-defined callback functions which are called during each iteration of the registration loop and which provide data corresponding to that iteration, such as the change in the transformation parameters, the current value of the registration error function, etc. This callback feature is useful

for implementing routines that perform tasks such as printing the iteration data to a terminal or logging the iteration data within a file for later analysis. The implementation of the callback feature is an example of the observer design pattern.[132] At the end of each iteration, a range of termination conditions are checked, including minimal thresholds on the change in the transformation parameters, minimal thresholds on the value and percentage of change in the registration error function, and a maximum iteration count threshold. In the studies conducted for this dissertation, the registration error thresholds were not used (i.e., they were set to zero), and algorithm termination was based on the change in the transformation parameters and the maximum iteration count. Support is also provided for algorithm-specific termination conditions, such as the technique used by the IMLP algorithm to detect cycling.

The generic routine that manages the PD-tree-search procedure is described by Algorithm 10.2. In similar manner to the generic registration loop routine, the generic PD-tree search routine takes a PD-tree algorithm object as input in order to customize the search behavior to each type of algorithm. The search begins by computing the match error for a candidate match point that was also provided as an input. Then a recursive node search procedure is started by calling the "NodeSearch" routine from the root of the tree and working down to the leaf nodes (as an optimization, the search could be started from the two children of the root, saving one node-boundary test on the root node since all datums must lie within the root; this of course implies that children of the root node must actually exist). It is within the NodeSearch routine

Algorithm 10.1. Generic Registration Loop Routine

input : ICP algorithm object: alg
Optimization options object: opt
Callback functions: callbacks
Initial guess for the transformation parameters: $\mathbf{T}_{\text{guess}}$
output: Registration solution: $\mathbf{T}$

```
// initialize
T ← T_guess
iteration ← 0 ,   terminate ← false ,   termCount ← 0
alg.Initialize( T )
// registration loop
while terminate == false do
    iteration ← iteration + 1
    // correspondence phase
    alg.ComputeMatches()
    if iteration == 1 then
        E ← alg.EvaluateErrorFunction()
        foreach callback function do
            Invoke the callback function; send data from this iteration as input
        end
    end
    // registration phase
    T_prior ← T
    T ← alg.RegisterMatches()
    ΔT = [ΔR, Δt] ← T ∘ T_prior^-1 ,   Δa ← Rodrigues form of ΔR
    E_prior ← E
    E ← alg.EvaluateErrorFunction()
    ΔE ← (E − E_prior)/E_prior
    // callbacks
    foreach callback function do
        Invoke the callback function; send data from this iteration as input
    end
    // termination test
    if alg.Terminate( T ) == true then terminate ← true
    if (‖Δa‖ < opt.Δα_min and ‖Δt‖ < opt.Δt_min) or
       E < opt.E_min or ΔE < opt.E_tol or iteration ≥ opt.maxIter then
        termCount ← termCount + 1
        if termCount ≥ opt.termHoldCount then terminate ← true
    else
        termCount ← 0
    end
end
```

that the customization by algorithm type occurs. The first step in the NodeSearch routine is to compare the bounds of the node with the current best match error by invoking the PD-tree interface method "NodePossiblyCloser()" on the algorithm object. If this method call returns false, then no better match within the node is possible, and the NodeSearch routine returns with no change to the best match. Otherwise, the node search continues. If the node being searched is a leaf node, then the match error is computed for each datum in the node by invoking the PD-tree interface method "ComputeBestMatchOnDatum()" on the algorithm object. If the node being searched is not a leaf node, then the node search continues by recursively calling the NodeSearch routine on the two child nodes.

In practice, the PD-tree interface object that is passed to the generic PD-tree search routine is made to be part of the ICP interface object. This enables the complete algorithm-specific code for each algorithm to be contained within the one algorithm object that is passed to the generic ICP registration routine. The PD-tree interface object can be combined with the ICP interface object using one of two techniques. One technique is to use multiple inheritance (which C++ supports) in order to include the methods for both the ICP interface and for the PD-tree interface directly within the same class. The other technique is to implement the PD-tree interface in an independent class, and then include an instance of the PD-tree interface class via composition within the object that implements the ICP interface. Since the generic PD-tree search routine is called from the "ComputeMatches" method of the

ICP interface, each algorithm maintains its own control over how this is implemented and either technique could be used.

Algorithm 10.2. Generic PD-Tree Search Routine

input : PD-tree algorithm object: alg
Data feature being matched: $\mathbf{x}$
Current candidate for best match: $\mathbf{y}_{\text{best}}$
output: Final best match: $\mathbf{y}_{\text{best}}$

```
1  E_best ← alg.ComputeBestMatchOnDatum( x, y_best )
2  [ y_best, E_best ] ← NodeSearch( alg, PDTree.Root, x, y_best, E_best )
3  Return y_best

   Procedure NodeSearch( alg, node, x, y_best, E_best )
4      if alg.NodePossiblyCloser( node, x, E_best ) == true then
5          if node is a leaf then
6              foreach datum ∈ node do
7                  [ y_datum, E_datum ] ← alg.ComputeBestMatchOnDatum(x, datum)
8                  if E_datum < E_best then [ y_best, E_best ] ← [ y_datum, E_datum ]
9              end
10         else
11             [ y_LChild, E_LChild] ← NodeSearch(alg, node.leftChild, x, y_best, E_best)
12             if E_LChild < E_best then [ y_best, E_best ] ← [ y_LChild, E_LChild ]
13             [ y_RChild, E_RChild] ← NodeSearch(alg, node.rightChild, x, y_best, E_best)
14             if E_RChild < E_best then [ y_best, E_best ] ← [ y_RChild, E_RChild ]
15         end
16     end
17     Return [ y_best, E_best ]
```

10.2 Concluding Remarks

The software architecture described in this chapter greatly simplifies the process of developing new ICP-based algorithms by decoupling the algorithm-specific elements of the ICP-based procedure from the core framework. By using an abstract interface

to bridge interactions between the generic core framework and the code that is unique to each algorithm, new algorithms can be added and old algorithms can be modified with minimal effort and with no changes to the core framework.

The development process for a new algorithm is often an iterative one, going through many versions and alternative implementations while discovering the best methods. The naive approach for testing minor and even major variants of the same algorithm is to activate and deactivate snippets of code using comments and/or preprocessor directives. Discovering the best method generally requires validation, often by running an extensive set of test cases, such as simulations, to determine how the algorithm behaves under each condition. Using the naive approach to test each algorithm variant requires the user to restart the validation study following each code modification, which is both tedious and error prone. However, using the algorithm-class-based software architecture that is described in this chapter, different versions of the same algorithm can be developed in parallel; through judicious use of algorithm sub-classing, each variant can be implemented with minimal effort and without rewriting shared functionality. Validation testing for each algorithm variant can then be performed as a batch process by dynamically swapping out the algorithm objects and rerunning the same tests without any further user interaction. This is one example of how this software architecture supports not only efficient implementation but also efficient design and testing of new algorithms.

In addition to the C++ software architecture described in this chapter, a Matlab-

based software interface has also been developed that enables the registration algorithms to be launched from Matlab rather than from C++. The Matlab interface was created by writing a mirror implementation of the generic registration loop of Algorithm 10.1 within Matlab and calling the C++ code in order to invoke the algorithm-specific methods, which were compiled into unique mex files—one for each algorithm. Running the generic registration loop within Matlab provides the benefit of more open access to the registration data during each iteration of the registration, as well as provides access to the convenient visualization capabilities that Matlab has to offer, such as enabling plotting of the shape alignment as the registration iterates. The runtime overhead of using Matlab as a front-end interface to the mex-compiled C++ code is negligible, as all the heavy lifting of the correspondence and registration phases can be performed within C++. The Matlab interface also enables the flexibility of some (or even all) of the algorithm-specific methods to be implemented within Matlab, while other of the algorithm's methods may still be implemented within C++. In particular, this approach can be very helpful for the registration phase in order to take advantage of the optimization capabilities offered by Matlab's capable optimization toolbox. This is the approach that was taken when developing the code for the V-IMLOP algorithm (Chapter 8). The other algorithms developed in this dissertation were completely implemented in C++, and in some cases the Matlab interface was used as a front end for running the experiments.

10.3 Contributions

The contributions from the chapter include:

8. An extensible software architecture supporting rapid development of ICP-based registration algorithms

Appendix A

Notation

Following are the notation conventions that are used throughout this manuscript.

a	lowercase letters are scalars
$\mathbf{a}$	lowercase bold letters are vectors
$\hat{\mathbf{a}}$	lowercase bold letters with a hat symbol are unit vectors
$\mathbf{A}$	uppercase bold letters are matrices
$\mathbf{I}$	the identity matrix; if not explicitly stated, the dimensions of this matrix are determined from context
$\mathbf{x}_\mathrm{p}$	is a feature position
$\hat{\mathbf{x}}_\mathrm{n}$	is a feature orientation
$(\mathbf{x}_\mathrm{p}, \hat{\mathbf{x}}_\mathrm{n})$	is an oriented point, having both a position and orientation component
$(\mathbf{x}_\mathrm{3dp}, \hat{\mathbf{x}}_\mathrm{2dn})$	is a projection-oriented point, having both a 3D position, $\mathbf{x}_\mathrm{3dp}$, and a 2D orientation, $\hat{\mathbf{x}}_\mathrm{2dn}$, component
$\mathbf{x}$	is a data-shape point, which may be non-oriented, oriented, or projection-oriented
$\mathbf{y}$	is a model-shape point, which may be non-oriented, oriented, or projection-oriented

T is a transformation; depending on the context, **T** may represent:

- rigid-body transformation: consists of a rotation matrix, **R**, and a translation vector, **t**; this composition of parameters is occasionally made explicit in the text by the expression $\mathbf{T} = [\mathbf{R}, \mathbf{t}]$
- similarity transformation: consists of a rotation matrix, **R**, translation vector, **t**, and a scaling factor, s; this composition of parameters is occasionally made explicit in the text by the expression $\mathbf{T} = [s, \mathbf{R}, \mathbf{t}]$

$\mathrm{T}(\cdot)$ is the transformation operator; it transforms an input feature, **x**, by the current transformation, **T**; its definition varies both by the type of transformation that **T** currently represents (determined by context) and by the type of input feature. The various possibilities are defined below:

- rigid-body transformation:
  non-oriented input, $\mathbf{x} = \mathbf{x}_\mathrm{p}$: $\mathrm{T}(\mathbf{x}) = \mathbf{R}\mathbf{x} + \mathbf{t}$
  oriented input, $\mathbf{x} = (\mathbf{x}_\mathrm{p}, \hat{\mathbf{x}}_\mathrm{n})$: $\mathrm{T}(\mathbf{x}) = (\mathbf{R}\mathbf{x}_\mathrm{p} + \mathbf{t}, \mathbf{R}\hat{\mathbf{x}}_\mathrm{n})$
- similarity transformation:
  non-oriented input, $\mathbf{x} = \mathbf{x}_\mathrm{p}$: $\mathrm{T}(\mathbf{x}) = s\mathbf{R}\mathbf{x} + \mathbf{t}$
  oriented input, $\mathbf{x} = (\mathbf{x}_\mathrm{p}, \hat{\mathbf{x}}_\mathrm{n})$: $\mathrm{T}(\mathbf{x}) = (s\mathbf{R}\mathbf{x}_\mathrm{p} + \mathbf{t}, \mathbf{R}\hat{\mathbf{x}}_\mathrm{n})$

Appendix B

Computing the Most Likely Correspondence Point on a PD-Tree Datum

In this appendix we briefly discuss how to compute the point of most likely correspondence on a given datum of a PD tree under an anisotropic noise model, as needed for Step 6 of the IMLP node search algorithm (Algorithm 4.3). This procedure is also needed for similar steps within the node search procedures for the G-IMLOP, P-IMLOP, and V-IMLOP algorithms. The match error equations used in this appendix correspond to the match error equations for the IMLP algorithm. When applying this procedure for the G-IMLOP, P-IMLOP, and V-IMLOP aglorithms, the positional components of their match error equations should be used instead.

APPENDIX B. COMPUTING THE MOST LIKELY CORRESPONDENCE POINT ON A PD-TREE DATUM

When the target shape is represented by a point cloud, then computing the most likely correspondence point on a datum is trivial, since each datum consists of only a single point, $\mathbf{y}$. In this case, each datum has associated with it a noise-model covariance, $\mathbf{\Sigma}_{\mathrm{y}}$, that is stored in the PD tree alongside the datum point. The match error for the datum is then simply computed as

$$\begin{aligned} \mathrm{E}_{\mathrm{match}}(\mathbf{x}\,;\mathbf{y},\mathbf{\Sigma}_{\mathrm{x}},\mathbf{\Sigma}_{\mathrm{y}},\mathbf{R},\mathbf{t}) = \log\left|\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}}+\mathbf{\Sigma}_{\mathrm{y}}\right| \\ +(\mathbf{y}-\mathbf{R}\mathbf{x}-\mathbf{t})^{\mathsf{T}}(\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}}+\mathbf{\Sigma}_{\mathrm{y}})^{-1}(\mathbf{y}-\mathbf{R}\mathbf{x}-\mathbf{t}) \end{aligned} \quad \text{(B.1)}$$

where $\mathbf{R}$ and $\mathbf{t}$ are the current estimates of the transformation parameters and where $\mathbf{x}$ and $\mathbf{\Sigma}_{\mathrm{x}}$ represent the source point being matched and its associated noise-model covariance.

The problem is more complicated when the target shape is represented by a mesh, where each datum represents a single triangle in the mesh. For the experiments in this paper, we make the assumption that all points on a given triangle share a common noise model. Thus, a single covariance, $\mathbf{\Sigma}_{\mathrm{y}}$, is associated with each datum triangle and stored in the PD tree.

The most likely match on a triangular datum is computed by performing a transformation of the physical space in order to transform the ellipsoidal level sets of the match-error function to a new space where the match-error function has spherical level sets. In this new space, the best match is then simply computed as the closest

point on the transformed triangle to the data point, $\mathbf{x}$. The procedure is to first transform each vertex, $\mathbf{v_i}$, of the datum triangle to form a new triangle having vertices $\mathbf{v_i'}$

$$\mathbf{v_i'} = \mathbf{N}(\mathbf{v_i} - \mathbf{R}\mathbf{x} - \mathbf{t}) \quad \text{for } i = 1, \ldots, n \tag{B.2}$$

where $\mathbf{N}$ is defined as

$$\mathbf{N}\mathbf{N}^{\mathsf{T}} = (\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\mathrm{y}})^{-1}\,. \tag{B.3}$$

$\mathbf{N}$ may be obtained by computing the eigen decomposition $(\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}}\mathbf{R}^{\mathsf{T}}+\boldsymbol{\Sigma}_{\mathrm{y}})^{-1} = \mathbf{V}\mathbf{D}\mathbf{V}^{\mathsf{T}}$ and setting $\mathbf{N} = \mathbf{V}\mathbf{D}^{\frac{1}{2}}$. Note that very efficient (closed-form) solutions exist for computing the eigen decomposition of a 3×3 symmetric matrix; one implementation is provided by the WildMagic5 C++ library.[73] The second step of the procedure is to compute the closest point on the new triangle to the origin; we refer to this point as $\mathbf{y}'$. The final step is to transform $\mathbf{y}'$ back to the original space by applying the inverse operation of (B.2), which provides the most likely match, $\mathbf{y}$, for this datum. The match error of the datum is then computed by (B.1) using the most likely match point, $\mathbf{y}$, and the covariance, $\boldsymbol{\Sigma}_{\mathrm{y}}$, that is associated with the datum triangle.

Note that the G-IMLOP, P-IMLOP, and V-IMLOP algorithms assume no noise on the model shape; thus, for these algorithms, $\boldsymbol{\Sigma}_{\mathrm{y}}$ is set to zero as implied by the match error equations associated with these algorithms.

Appendix C

Equivalent Forms for the Generalized Total-Least-Squares (GTLS) Problem of Aligning Corresponding Point Sets

In this appendix, an equivalence is established between two alternate representations for the generalized total-least-squares (GTLS) problem of registering two corresponding point sets under a generalized noise model, which is the problem that is solved by the registration phase of the IMLP algorithm (Section 4.4). Our aim is to show that the unconstrained optimization (unconstrained subject to a valid rotation,

APPENDIX C. EQUIVALENT FORMS FOR THE GENERALIZED TOTAL-LEAST-SQUARES (GTLS) PROBLEM OF ALIGNING CORRESPONDING POINT SETS

that is)

$$\mathrm{E}_{\mathrm{GTLS}}(\mathbf{X}, \mathbf{Y}, \mathbf{\Sigma}_{\mathrm{X}}, \mathbf{\Sigma}_{\mathrm{Y}}) = \min_{[\mathbf{R}, \mathbf{t}]} \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t})^{\mathsf{T}} (\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}i}\mathbf{R}^{\mathsf{T}} + \mathbf{\Sigma}_{\mathrm{y}i})^{-1} (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}) \quad \text{(C.1)}$$

is equivalent to the constrained optimization

$$\mathrm{E}_{\mathrm{GTLS}}(\mathbf{X}, \mathbf{Y}, \mathbf{\Sigma}_{\mathrm{X}}, \mathbf{\Sigma}_{\mathrm{Y}}) = \min_{[\mathbf{R}, \mathbf{t}]} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}_i^*)^{\mathsf{T}} \mathbf{\Sigma}_{\mathrm{x}i}^{-1} (\mathbf{x}_i - \mathbf{x}_i^*) + \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{y}_i^*)^{\mathsf{T}} \mathbf{\Sigma}_{\mathrm{y}i}^{-1} (\mathbf{y}_i - \mathbf{y}_i^*)$$

$$\text{subject to: } \mathbf{y}_i^* = \mathbf{R}\mathbf{x}_i^* - \mathbf{t} \quad \text{(C.2)}$$

where $\mathbf{X} = \{\mathbf{x}_i\}$ and $\mathbf{Y} = \{\mathbf{y}_i\}$ are the measured data and model point sets that are in correspondence and $\mathbf{\Sigma}_{\mathrm{X}} = \{\mathbf{\Sigma}_{\mathrm{x}i}\}$ and $\mathbf{\Sigma}_{\mathrm{Y}} = \{\mathbf{\Sigma}_{\mathrm{y}i}\}$ are the noise covariances of these measured points. The point sets $\{\mathbf{x}_i^*\}$ and $\{\mathbf{y}_i^*\}$ represent the optimizer's estimates for the unknown, noise-free positions of the data and model points which, due to the correspondence assumption, are constrained to have perfect alignment under the transformation parameters, $\mathbf{R}$ and $\mathbf{t}$, that are being solved by the optimization.

To establish an equivalence between (C.1) and (C.2), we begin at (C.2) and derive expressions for the estimates of the noise-free point sets $\{\mathbf{x}_i^*\}$ and $\{\mathbf{y}_i^*\}$ in terms of the measured points, noise covariances, and transformation parameters. These expressions will then be substituted into the cost function of (C.2) which, through a subsequent series of algebraic simplifications, will be shown to be equivalent to the form of (C.1).

To begin, we solve for expressions of the noise-free estimates $\{\mathbf{x}_i^*\}$ and $\{\mathbf{y}_i^*\}$.

This may be accomplished using the method of Lagrange multipliers. Redefining the constraints as

$$\mathrm{F}_i(\mathbf{x}_i^*, \mathbf{y}_i^*, \mathbf{R}, \mathbf{t}) = \mathbf{y}_i^* - \mathbf{R}\mathbf{x}_i^* - \mathbf{t} = 0 \tag{C.3}$$

we obtain the following Lagrangian function

$$\mathcal{L} = \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}_i^*)^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{x}i}^{-1} (\mathbf{x}_i - \mathbf{x}_i^*) + \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{y}_i^*)^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{y}i}^{-1} (\mathbf{y}_i - \mathbf{y}_i^*) + \boldsymbol{\lambda}_i^{\mathsf{T}} \mathrm{F}_i(\mathbf{x}_i^*, \mathbf{y}_i^*, \mathbf{R}, \mathbf{t}) \tag{C.4}$$

which may be expressed in matrix form as

$$\mathcal{L} = (\mathbf{X} - \mathbf{X}^*)^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{X}}^{-1} (\mathbf{X} - \mathbf{X}^*) + (\mathbf{Y} - \mathbf{Y}^*)^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{Y}}^{-1} (\mathbf{Y} - \mathbf{Y}^*) + \boldsymbol{\lambda}^{\mathsf{T}} \mathbf{f} \tag{C.5}$$

where we have defined

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \quad \mathbf{X}^* = \begin{bmatrix} \mathbf{x}_1^* \\ \vdots \\ \mathbf{x}_n^* \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{bmatrix}, \quad \mathbf{Y}^* = \begin{bmatrix} \mathbf{y}_1^* \\ \vdots \\ \mathbf{y}_n^* \end{bmatrix},$$

$$\boldsymbol{\Sigma}_{\mathrm{X}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathrm{x}1} & & \\ & \ddots & \\ & & \boldsymbol{\Sigma}_{\mathrm{x}n} \end{bmatrix}, \quad \boldsymbol{\Sigma}_{\mathrm{Y}} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathrm{y}1} & & \\ & \ddots & \\ & & \boldsymbol{\Sigma}_{\mathrm{y}n} \end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \vdots \\ \boldsymbol{\lambda}_n \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} \mathrm{F}_i(\mathbf{x}_1^*, \mathbf{y}_1^*, \mathbf{R}, \mathbf{t}) \\ \vdots \\ \mathrm{F}_i(\mathbf{x}_n^*, \mathbf{y}_n^*, \mathbf{R}, \mathbf{t}\,) \end{bmatrix}.$$

The next step is to minimize the Lagrangian function with respect to the estimates of the noise-free point sets. This is done by solving for the partial derivatives of the

Lagrangian function with respect to each estimate

$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}^*} = -2\boldsymbol{\Sigma}_{\mathrm{X}}^{-1}(\mathbf{X} - \mathbf{X}^*) - \mathrm{diag}(\mathbf{R}^{\mathsf{T}})\boldsymbol{\lambda} \tag{C.6}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Y}^*} = -2\boldsymbol{\Sigma}_{\mathrm{Y}}^{-1}(\mathbf{Y} - \mathbf{Y}^*) + \boldsymbol{\lambda} \tag{C.7}$$

and setting the partial derivatives to zero, which produces the following equations

$$(\mathbf{X} - \mathbf{X}^*) = -\frac{1}{2}\boldsymbol{\Sigma}_{\mathrm{X}}\,\mathrm{diag}(\mathbf{R}^{\mathsf{T}})\boldsymbol{\lambda} \tag{C.8}$$

$$(\mathbf{Y} - \mathbf{Y}^*) = \frac{1}{2}\boldsymbol{\Sigma}_{\mathrm{Y}}\boldsymbol{\lambda}\,. \tag{C.9}$$

We now have all the equations required to solve for expressions of $\mathbf{x}_i^*$ and $\mathbf{y}_i^*$ in terms of the other parameters. Solving (C.9) for $\boldsymbol{\lambda}_i$

$$\boldsymbol{\lambda}_i = 2\boldsymbol{\Sigma}_{\mathrm{y}i}^{-1}(\mathbf{y}_i - \mathbf{y}_i^*) \tag{C.10}$$

and substituting into the relevant sub-component of (C.8) we obtain

$$(\mathbf{x}_i - \mathbf{x}_i^*) = -\boldsymbol{\Sigma}_{\mathrm{x}i}\mathbf{R}^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathrm{y}i}^{-1}(\mathbf{y}_i - \mathbf{y}_i^*)\,. \tag{C.11}$$

Substituting the constraint from (C.2) into (C.11) we have

$$(\mathbf{x}_i - \mathbf{x}_i^*) = -\boldsymbol{\Sigma}_{\mathrm{x}i}\mathbf{R}^{\mathsf{T}}\boldsymbol{\Sigma}_{\mathrm{y}i}^{-1}(\mathbf{y}_i - \mathbf{R}\mathbf{x}_i^* - \mathbf{t})\,. \tag{C.12}$$

Rearrangement of (C.12) produces the following expression for $\mathbf{x}_i^*$

$$\mathbf{x}_i^* = \left(\mathbf{I} + \boldsymbol{\Sigma}_{\mathrm{x}i}\mathbf{R}^\mathsf{T}\boldsymbol{\Sigma}_{\mathrm{y}i}^{-1}\mathbf{R}\right)^{-1}\left(\mathbf{x}_i + \boldsymbol{\Sigma}_{\mathrm{x}i}\mathbf{R}^\mathsf{T}\boldsymbol{\Sigma}_{\mathrm{y}i}^{-1}\left(\mathbf{y}_i - \mathbf{t}\right)\right) \tag{C.13}$$

$$= \mathbf{x}_i + \boldsymbol{\Sigma}_{\mathrm{x}i}\mathbf{R}^\mathsf{T}\left(\boldsymbol{\Sigma}_{\mathrm{y}i} + \mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}i}\mathbf{R}^\mathsf{T}\right)^{-1}\left(\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\right) . \tag{C.14}$$

The derivation of (C.14) from (C.13) is accomplished by expanding the multiplication with the inverse expression and then applying the following helpful identities

$$(\mathbf{I} + \mathbf{AB})^{-1} = \mathbf{I} - \mathbf{A}(\mathbf{BA} + \mathbf{I})^{-1}\mathbf{B} \tag{C.15}$$

$$(\mathbf{A} + \mathbf{B})^{-1}\mathbf{C} = (\mathbf{C}^{-1}\mathbf{A} + \mathbf{C}^{-1}\mathbf{B})^{-1} \tag{C.16}$$

$$\mathbf{C}(\mathbf{A} + \mathbf{B})^{-1} = (\mathbf{AC}^{-1} + \mathbf{BC}^{-1})^{-1} . \tag{C.17}$$

The identity of (C.15) follows as a simplification of

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1})^{-1}\mathbf{DA}^{-1} \tag{C.18}$$

which is described in [133], while those of (C.16) and (C.17) are easily verified.

An expression for $\mathbf{y}_i$ is obtained in similar manner by solving (C.8) for $\boldsymbol{\lambda}_i$ and substituting into (C.9) along with the constraint of (C.2), which leads to the equations

$$\mathbf{y}_i^* = \left(\mathbf{I} + \boldsymbol{\Sigma}_{\mathrm{y}i}\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}i}^{-1}\mathbf{R}^\mathsf{T}\right)^{-1}\left(\mathbf{y}_i + \boldsymbol{\Sigma}_{\mathrm{y}i}\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}i}^{-1}\left(\mathbf{x}_i + \mathbf{R}^\mathsf{T}\mathbf{t}\right)\right) \tag{C.19}$$

$$= \mathbf{y}_i - \boldsymbol{\Sigma}_{\mathrm{y}i}\left(\mathbf{R}\boldsymbol{\Sigma}_{\mathrm{x}i}\mathbf{R}^\mathsf{T} + \boldsymbol{\Sigma}_{\mathrm{y}i}\right)^{-1}\left(\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\right) . \tag{C.20}$$

The next step is to substitute the expressions of $\mathbf{x}_i^*$ from (C.14) and of $\mathbf{y}_i^*$ from (C.20) into the cost function of (C.2). To simplify the resulting equations, we make the following definitions

$$\mathbf{\Sigma}_i = (\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}i}\mathbf{R}^\mathsf{T} + \mathbf{\Sigma}_{\mathrm{y}i})$$

$$\mathbf{d}_i = \mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t}\,.$$

Following substitution of these definitions, the two terms within the cost function of (C.2) become

$$(\mathbf{x}_i - \mathbf{x}_i^*)^\mathsf{T}\mathbf{\Sigma}_{\mathrm{x}i}^{-1}(\mathbf{x}_i - \mathbf{x}_i^*) = \mathbf{d}_i^\mathsf{T}\mathbf{\Sigma}_i^{-1}\mathbf{R}\mathbf{\Sigma}_{\mathrm{x}i}\mathbf{R}^\mathsf{T}\mathbf{\Sigma}_i^{-1}\mathbf{d}_i \tag{C.21}$$

$$(\mathbf{y}_i - \mathbf{y}_i^*)^\mathsf{T}\mathbf{\Sigma}_{\mathrm{y}i}^{-1}(\mathbf{y}_i - \mathbf{y}_i^*) = \mathbf{d}_i^\mathsf{T}\mathbf{\Sigma}_i^{-1}\mathbf{\Sigma}_{\mathrm{y}i}\mathbf{\Sigma}_i^{-1}\mathbf{d}_i\,. \tag{C.22}$$

Substituting these equations into the cost function of (C.2), we have

$$
\begin{aligned}
\mathrm{E}_{\mathrm{GTLS}}(\mathbf{X}, \mathbf{Y}, \boldsymbol{\Sigma}_{\mathrm{X}}, \boldsymbol{\Sigma}_{\mathrm{Y}}) &= \min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{n} \left[(\mathbf{x}_i - \mathbf{x}_i^*)^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{x}i}^{-1} (\mathbf{x}_i - \mathbf{x}_i^*) + (\mathbf{y}_i - \mathbf{y}_i^*)^{\mathsf{T}} \boldsymbol{\Sigma}_{\mathrm{y}i}^{-1} (\mathbf{y}_i - \mathbf{y}_i^*) \right] \\
&\qquad\qquad \text{subject to: } \mathbf{y}_i^* = \mathbf{R}\mathbf{x}_i^* - \mathbf{t} \\
&= \min_{[\mathbf{R},\,\mathbf{t}]} \sum_{i=1}^{n} \left[\mathbf{d}_i^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1} \mathbf{R} \boldsymbol{\Sigma}_{\mathrm{x}i} \mathbf{R}^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1} \mathbf{d}_i + \mathbf{d}_i^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\Sigma}_{\mathrm{y}i} \boldsymbol{\Sigma}_i^{-1} \mathbf{d}_i \right] \\
&= \min_{[\mathbf{R},\,\mathbf{t}]} \sum_{i=1}^{n} \mathbf{d}_i^{\mathsf{T}} \left(\boldsymbol{\Sigma}_i^{-1} \mathbf{R} \boldsymbol{\Sigma}_{\mathrm{x}i} \mathbf{R}^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1} + \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\Sigma}_{\mathrm{y}i} \boldsymbol{\Sigma}_i^{-1} \right) \mathbf{d}_i \\
&= \min_{[\mathbf{R},\,\mathbf{t}]} \sum_{i=1}^{n} \mathbf{d}_i^{\mathsf{T}} \left(\boldsymbol{\Sigma}_i^{-1} \left(\mathbf{R} \boldsymbol{\Sigma}_{\mathrm{x}i} \mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\mathrm{y}i} \right) \boldsymbol{\Sigma}_i^{-1} \right) \mathbf{d}_i \\
&= \min_{[\mathbf{R},\,\mathbf{t}]} \sum_{i=1}^{n} \mathbf{d}_i^{\mathsf{T}} \left(\boldsymbol{\Sigma}_i^{-1} \boldsymbol{\Sigma}_i \boldsymbol{\Sigma}_i^{-1} \right) \mathbf{d}_i \\
&= \min_{[\mathbf{R},\,\mathbf{t}]} \sum_{i=1}^{n} \mathbf{d}_i^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1} \mathbf{d}_i \\
&= \min_{[\mathbf{R},\,\mathbf{t}]} \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t})^{\mathsf{T}} \left(\mathbf{R} \boldsymbol{\Sigma}_{\mathrm{x}i} \mathbf{R}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\mathrm{y}i} \right)^{-1} (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i - \mathbf{t})
\end{aligned}
\tag{C.23}
$$

where the final form of (C.23) is equivalent to (C.1), which completes the derivation of equivalence between (C.1) and (C.2).

Appendix D

Samples from the Fisher and Kent Distributions

In this appendix, the Fisher and Kent directional distributions are defined, and samples from the Fisher and Kent distributions are provided in order to illustrate the effects of the parameters of each distribution.

For the isotropic Fisher distribution, whose PDF is defined as

$$f(\hat{\mathbf{x}}_{\mathrm{n}}\,;\,\hat{\mu},\kappa) = \frac{\kappa}{2\pi(e^{\kappa}-e^{-\kappa})}e^{\kappa\hat{\mu}^{\mathsf{T}}\hat{\mathbf{x}}_{\mathrm{n}}} \tag{D.1}$$

the distribution parameters include the *concentration*, κ, and the *central direction*, $\hat{\mu}$.

For the anisotropic Kent distribution, whose PDF is defined as

$$f(\hat{\mathbf{x}}_{\mathrm{n}}\,;\,\hat{\mu},\kappa,\beta,\hat{\boldsymbol{\gamma}}_1,\hat{\boldsymbol{\gamma}}_2) = \frac{1}{\mathrm{c}(\kappa,\beta)} e^{\kappa\hat{\mu}^{\mathsf{T}}\hat{\mathbf{x}}_{\mathrm{n}}+\beta\left((\hat{\boldsymbol{\gamma}}_1^{\mathsf{T}}\hat{\mathbf{x}}_{\mathrm{n}})^2-(\hat{\boldsymbol{\gamma}}_2^{\mathsf{T}}\hat{\mathbf{x}}_{\mathrm{n}})^2\right)} \tag{D.2}$$

the distribution parameters include, in addition to those of the Fisher distribution, the *ellipticity*, β ($0 \leq 2\beta < \kappa$), and the major and minor axes, $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$, respectively. In the following illustrations, the *eccentricity*, e, is used in place of the ellipticity, β, which is defined as

$$e = \frac{2\beta}{\kappa} \tag{D.3}$$

and which has values on the interval $[0, 1)$.

Figure D.1 illustrates six different examples of Fisher and Kent distributions on the unit sphere, each consisting of 200 samples. For each plot, the central direction, $\hat{\mu}$, is oriented towards the pole. The plots are generated using the wrapped-Gaussian approximation to the Fisher and Kent distributions.[85] Each plot was generated from the same random data that has been refitted to the parameters of each example distribution.

The upper three plots have eccentricities of zero, and therefore represent isotropic Fisher distributions. These plots illustrate the effect of the concentration parameter, κ. Notice that as κ decreases in value, the distribution of the orientations becomes more widely dispersed.

The lower three plots represent anisotropic Kent distributions and demonstrate the

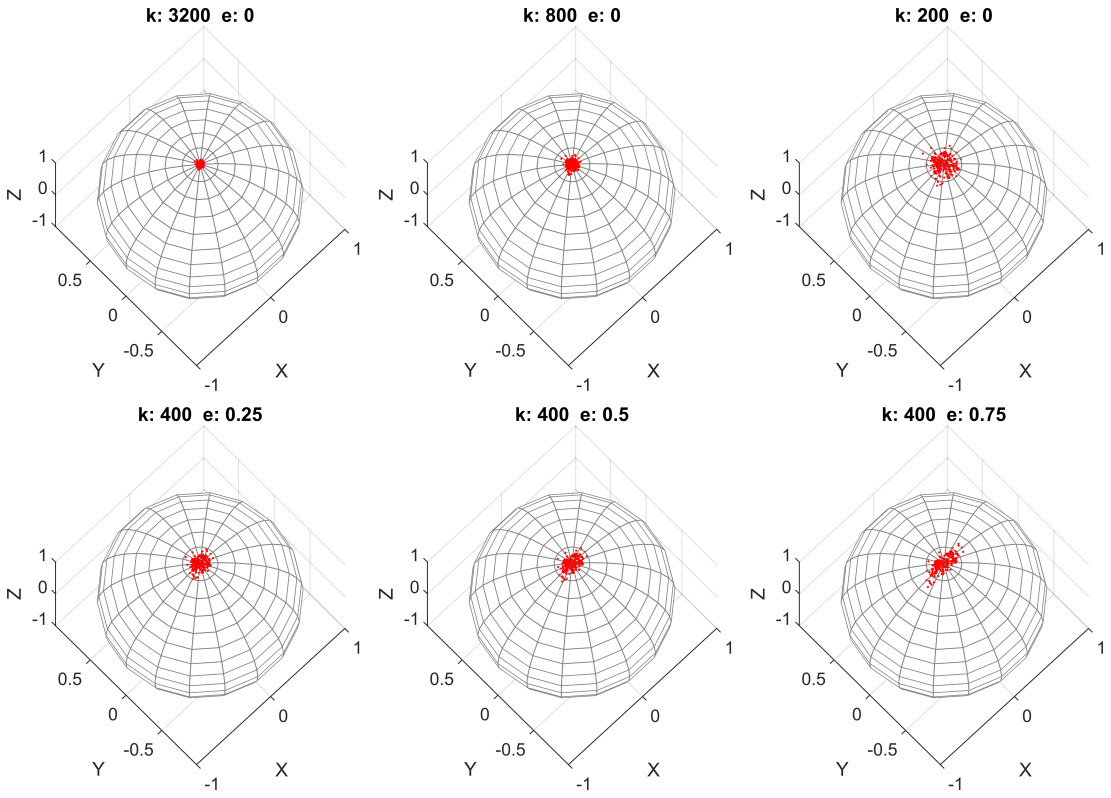

Figure D.1: Example Fisher (top row) and Kent (bottom row) distributions; each plot represents the same underlying data that has been refitted to the parameters of each example distribution. One set of 200 samples was drawn from a standard normal distribution. This single set of normally distributed samples was then converted (by scaling relative to the appropriate covariance matrix) into each of the non-standard normal distributions that approximate the Fisher and Kent distributions illustrated in this figure.

effect of the eccentricity, e, for a fixed concentration, κ, with the major and minor axes oriented along the x- and y-axis, respectively. Note that as the eccentricity increases, the distribution of the orientations becomes more widely dispersed along the major axis and less widely dispersed along the minor axis.

Appendix E

The G-IMLOP Match Error Function

In this appendix, it is shown that the G-IMLOP algorithm's match error function

$$\mathrm{E}_{\text{G-IMLOP}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\Sigma}, \kappa, \beta, \hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\gamma}}_2) = \frac{1}{2}(\mathbf{y}_\mathrm{p} - \mathbf{x}_\mathrm{p})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{y}_\mathrm{p} - \mathbf{x}_\mathrm{p}) + \kappa(1 - \hat{\mathbf{y}}_\mathrm{n}^\mathsf{T}\hat{\mathbf{x}}_\mathrm{n}) - \beta\left((\hat{\boldsymbol{\gamma}}_1^\mathsf{T}\hat{\mathbf{y}}_\mathrm{n})^2 - (\hat{\boldsymbol{\gamma}}_2^\mathsf{T}\hat{\mathbf{y}}_\mathrm{n})^2\right) \quad \text{(E.1)}$$

is always positive, where in (E.1) the simplification has been made, without loss of generality, to disregard the transformation, $\mathbf{T} = [\mathbf{R}, \mathbf{t}]$, of the oriented data point, $\mathbf{x} = (\mathbf{x}_\mathrm{p}, \hat{\mathbf{x}}_\mathrm{n})$. To demonstrate that this match error function is always positive, it suffices to show that each term is always positive. The match error function is thus

broken into two terms: a *position* term

$$E_{\text{pos}} = \frac{1}{2}(\mathbf{y}_{\text{p}} - \mathbf{x}_{\text{p}})^{\mathsf{T}}\Sigma^{-1}(\mathbf{y}_{\text{p}} - \mathbf{x}_{\text{p}}) \tag{E.2}$$

and an *orientation* term

$$E_{\text{orn}} = \kappa(1 - \hat{\mathbf{y}}_{\text{n}}^{\mathsf{T}}\hat{\mathbf{x}}_{\text{n}}) - \beta\left((\hat{\boldsymbol{\gamma}}_1^{\mathsf{T}}\hat{\mathbf{y}}_{\text{n}})^2 - (\hat{\boldsymbol{\gamma}}_2^{\mathsf{T}}\hat{\mathbf{y}}_{\text{n}})^2\right) \quad . \tag{E.3}$$

Regarding the position term, note that Σ is a positive-definite covariance matrix; thus, the quadratic position term must be greater than or equal to zero.[134]

Regarding the orientation term, consider the probability density function (PDF) of the Kent distribution, which is defined as

$$f(\hat{\mathbf{x}}_{\text{n}} \,|\, \hat{\mathbf{y}}_{\text{n}}, \kappa, \beta, \hat{\boldsymbol{\gamma}}_1, \hat{\boldsymbol{\gamma}}_2) = \frac{1}{\text{c}(\kappa, \beta)} e^{\kappa\hat{\mathbf{y}}_{\text{n}}^{\mathsf{T}}\hat{\mathbf{x}}_{\text{n}} + \beta\left((\hat{\boldsymbol{\gamma}}_1^{\mathsf{T}}\hat{\mathbf{x}}_{\text{n}})^2 - (\hat{\boldsymbol{\gamma}}_2^{\mathsf{T}}\hat{\mathbf{x}}_{\text{n}})^2\right)} \quad . \tag{E.4}$$

It is well known that the PDF of the Kent distribution attains its maximum value when $\hat{\mathbf{x}}_{\text{n}}$ is oriented along the central direction, $\hat{\mathbf{y}}_{\text{n}}$.[85] Since maximizing the exponential term also maximizes the PDF, it follows that

$$\kappa\hat{\mathbf{y}}_{\text{n}}^{\mathsf{T}}\hat{\mathbf{x}}_{\text{n}} + \beta\left((\hat{\boldsymbol{\gamma}}_1^{\mathsf{T}}\hat{\mathbf{x}}_{\text{n}})^2 - (\hat{\boldsymbol{\gamma}}_2^{\mathsf{T}}\hat{\mathbf{x}}_{\text{n}})^2\right) \leq \kappa\hat{\mathbf{y}}_{\text{n}}^{\mathsf{T}}\hat{\mathbf{y}}_{\text{n}} + \beta\left((\hat{\boldsymbol{\gamma}}_1^{\mathsf{T}}\hat{\mathbf{y}}_{\text{n}})^2 - (\hat{\boldsymbol{\gamma}}_2^{\mathsf{T}}\hat{\mathbf{y}}_{\text{n}})^2\right) = \kappa \quad . \tag{E.5}$$

The simplification of the right-hand side of the inequality to κ follows because the

central direction, $\hat{\mathbf{y}}_{\mathrm{n}}$, is a unit vector and is perpendicular to the major and minor axes, $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$. Concerning the redefinition of $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$ such that these axes are perpendicular to $\hat{\mathbf{x}}_{\mathrm{n}}$ rather than perpendicular to $\hat{\mathbf{y}}_{\mathrm{n}}$, as described in Section 6.1 and as reflected in Equation (E.1), this simplification still holds in this case because $\hat{\mathbf{x}}_{\mathrm{n}}$ would be parallel to $\hat{\mathbf{y}}_{\mathrm{n}}$, and thus $\hat{\mathbf{y}}_{\mathrm{n}}$ would be perpendicular to $\hat{\boldsymbol{\gamma}}_1$ and $\hat{\boldsymbol{\gamma}}_2$.

Applying the inequality of (E.5) to the orientation term of (E.3) results in

$$E_{\mathrm{orn}} = \kappa(1 - \hat{\mathbf{y}}_{\mathrm{n}}^{\mathsf{T}}\hat{\mathbf{x}}_{\mathrm{n}}) - \beta\left((\hat{\boldsymbol{\gamma}}_1^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}})^2 - (\hat{\boldsymbol{\gamma}}_2^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}})^2\right) \tag{E.6}$$

$$= \kappa - \left[\kappa\hat{\mathbf{y}}_{\mathrm{n}}^{\mathsf{T}}\hat{\mathbf{x}}_{\mathrm{n}} + \beta\left((\hat{\boldsymbol{\gamma}}_1^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}})^2 - (\hat{\boldsymbol{\gamma}}_2^{\mathsf{T}}\hat{\mathbf{y}}_{\mathrm{n}})^2\right)\right] \tag{E.7}$$

$$\geq \kappa - \kappa = 0 \ . \tag{E.8}$$

Thus, we have shown that the orientation term is greater than or equal to zero.

Since both the position and orientation terms have been shown to be greater than or equal to zero, it follows that the match error function of (E.1) is always positive.

Appendix F

Rodrigues-Based Partial Derivatives of the 3×3 Rotation Matrix

The Rodrigues parameterization for a rotation in 3D space represents rotation as a 3-vector, $\boldsymbol{a} = [a_x, a_y, a_z]$, whose direction and magnitude signify the axis and angular extent of the rotation, respectively. The notation $\mathrm{R}(\boldsymbol{a})$ is used to signify the 3×3 rotation matrix corresponding to the Rodrigues vector, $\boldsymbol{a}$, which is defined as

$$\mathrm{R}(\boldsymbol{a}) = \mathbf{I} + \sin(\theta)\,\mathrm{skew}(\boldsymbol{\alpha}) + (1 - \cos(\theta))(\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathsf{T}} - \mathbf{I}) \tag{F.1}$$

$$\theta = \|\boldsymbol{a}\|_2 \tag{F.2}$$

$$\boldsymbol{\alpha} = \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|} \tag{F.3}$$

where θ is the magnitude of $\boldsymbol{a}$, representing the angle of rotation, and $\boldsymbol{\alpha}$ is the unit vector in the direction of $\boldsymbol{a}$, representing the axis of rotation. The expression, skew($\boldsymbol{\alpha}$), is the skew-symmetric matrix formed from the elements of $\boldsymbol{\alpha}$. Note that the formula of Equation (F.1) is equivalent to the prior Rodrigues formula given in Equation (6.7), with the difference being that here the expression, skew$(\boldsymbol{\alpha})^2$, has been written in the expanded form, $(\boldsymbol{\alpha}\boldsymbol{\alpha}^\mathsf{T} - \mathbf{I})$.

The purpose of this appendix is to define the partial derivatives of the 3×3 rotation matrix, R($\boldsymbol{a}$), with respect to each element of the Rodrigues vector, $\boldsymbol{a}$. In the following definition, the notation dR$(\boldsymbol{a}, \boldsymbol{da})$ is used to represent the first-order change in the values of the elements of the rotation matrix, R($\boldsymbol{a}$), with respect to a change of $\boldsymbol{da}$ in the value of the Rodrigues vector, $\boldsymbol{a}$. Thus, the partial derivatives of R($\boldsymbol{a}$) with respect to the x, y, and z elements of $\boldsymbol{a}$ will be finally obtained by using the notation, dR$(\boldsymbol{a}, \boldsymbol{da})$, and setting $\boldsymbol{da}$ equal to unit vectors oriented along the positive x, y, and z axes, respectively.

The expression for dR$(\boldsymbol{a}, \boldsymbol{da})$ is derived by applying the product rule for differentiation to Equation (F.1), which produces in the following expression

$$\begin{aligned} \mathrm{dR}(\boldsymbol{a}, \boldsymbol{da}) &= \cos(\theta) d\theta \,\mathrm{skew}(\boldsymbol{\alpha}) + \sin(\theta)\frac{1}{\theta^2}\,\mathrm{skew}(\theta \boldsymbol{da} - d\theta \boldsymbol{a}) \\ &+ \sin(\theta) d\theta(\boldsymbol{\alpha}\boldsymbol{\alpha}^\mathsf{T} - \mathbf{I}) + (1 - \cos(\theta))\frac{1}{\theta^3}\left(\theta(\boldsymbol{da}\,\boldsymbol{a}^\mathsf{T} + \boldsymbol{a}\,\boldsymbol{da}^\mathsf{T}) - 2d\theta \boldsymbol{a}\,\boldsymbol{a}^\mathsf{T}\right) \end{aligned} \tag{F.4}$$

$$d\theta = \boldsymbol{\alpha}^\mathsf{T} \boldsymbol{da} \ . \tag{F.5}$$

Finally, substituting the appropriate values for $\boldsymbol{da}$ into the expression, $\mathrm{dR}(\boldsymbol{a}, \boldsymbol{da})$, provides the following partial derivative equations for $\mathrm{R}(\boldsymbol{a})$, where each partial derivative is represented as a separate 3×3 matrix containing the partial derivative of each element of $\mathrm{R}(\boldsymbol{a})$ with respect to a single element of $\boldsymbol{a}$.

$$\frac{\partial \mathrm{R}(\boldsymbol{a})}{\partial a_x} = \mathrm{dR}(\boldsymbol{a}, \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathsf{T}}) \tag{F.6}$$

$$\frac{\partial \mathrm{R}(\boldsymbol{a})}{\partial a_y} = \mathrm{dR}(\boldsymbol{a}, \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^{\mathsf{T}}) \tag{F.7}$$

$$\frac{\partial \mathrm{R}(\boldsymbol{a})}{\partial a_z} = \mathrm{dR}(\boldsymbol{a}, \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathsf{T}}) \tag{F.8}$$

Note that special handling is required for the case of computing the partial derivatives when beginning from $\boldsymbol{a} = 0$, i.e., for the case of computing the partial derivatives of the identity rotation matrix. In this case, the rotation axis, $\boldsymbol{\alpha}$, is undefined, since the magnitude of the Rodrigues vector, $\boldsymbol{a}$, is equal to zero. This special case may be handled by setting the axis, $\boldsymbol{\alpha}$, of the rotation in the same direction as the differential vector, $\boldsymbol{da}$. Since θ is equal to zero in this case, the 2nd and 3rd terms of Equation (F.4) each evaluate to $0/0$, which should be specially interpreted as being equal to zero in order to prevent division by zero. Applying this special handling reduces the partial derivatives of the identity matrix to being simply equal to $\mathrm{skew}(\boldsymbol{da})$. In other words, $\mathrm{dR}(\boldsymbol{a}, \boldsymbol{da})$ should be interpreted as being equal to $\mathrm{skew}(\boldsymbol{da})$ whenever $\boldsymbol{a}$ is very near to zero.

Bibliography

[1] W. R. Crum, T. Hartkens, and D. L. G. Hill, "Non-rigid image registration: Theory and practice," *The British Journal of Radiology*, vol. 77, no. suppl_2, pp. S140–S153, 2004. [Online]. Available: http://dx.doi.org/10.1259/bjr/25329214

[2] S. Cohan, "ROBODOC achieves pinless registration," *Industrial Robot: An International Journal*, vol. 28, no. 5, pp. 381–386, 2001. [Online]. Available: http://dx.doi.org/10.1108/01439910110401277

[3] J. B. A. Maintz and M. A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, no. 1, pp. 1–36, 1998. [Online]. Available: http://dx.doi.org/10.1016/S1361-8415(01)80026-8

[4] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992. [Online]. Available: http://dx.doi.org/10.1109/34.121791

[5] G. C. Sharp, S. W. Lee, and D. K. Wehe, "ICP registration using invariant features," *IEEE Transactions on Pattern Analysis and Machine*

Intelligence, vol. 24, no. 1, pp. 90–102, 2002. [Online]. Available: http://dx.doi.org/10.1109/34.982886

[6] K. Pulli and M. Pietikäinen, "Range image segmentation based on decomposition of surface normals," in *Proceedings of the 8th Scandinavian Conference on Image Analysis*, vol. 2, 1993, pp. 893–899.

[7] C. Schutz, T. Jost, and H. Hugli, "Multi-feature matching algorithm for free-form 3D surface registration," in *Fourteenth International Conference on Pattern Recognition, 1998*, vol. 2, Aug. 1998, pp. 982–984. [Online]. Available: http://dx.doi.org/10.1109/ICPR.1998.711852

[8] A. E. Johnson and M. Hebert, "Surface registration by matching oriented points," in *3-D Digital Imaging and Modeling, 1997. Proceedings., International Conference on Recent Advances in.* IEEE, 1997, pp. 121–128. [Online]. Available: http://dx.doi.org/10.1109/IM.1997.603857

[9] D. L. G. Hill, P. G. Batchelor, M. Holden, and D. J. Hawkes, "Medical image registration," *Physics in Medicine and Biology*, vol. 46, no. 3, p. R1, 2001. [Online]. Available: http://dx.doi.org/10.1088/0031-9155/46/3/201

[10] S. Billings, A. Kapoor, M. Keil, B. J. Wood, and E. Boctor, "A hybrid surface/image-based approach to facilitate ultrasound/CT registration," in *SPIE, Medical Imaging 2011: Ultrasonic Imaging, Tomography, and Therapy*,

J. D'hooge and M. M. Doyley, Eds., vol. 7968, 2011, p. 79680V. [Online]. Available: http://dx.doi.org/10.1117/12.878941

[11] D. M. Cash, T. K. Sinha, W. C. Chapman, H. Terawaki, B. M. Dawant, R. L. Galloway, and M. I. Miga, "Incorporation of a laser range scanner into image-guided liver surgery: Surface acquisition, registration, and tracking," *Medical Physics*, vol. 30, no. 7, pp. 1671–1682, 2003. [Online]. Available: http://dx.doi.org/10.1118/1.1578911

[12] T. P. Rauth, P. Q. Bao, R. L. Galloway, J. Bieszczad, E. M. Friets, D. A. Knaus, D. B. Kynor, and A. J. Herline, "Laparoscopic surface scanning and subsurface targeting: Implications for image-guided laparoscopic liver surgery," *Surgery*, vol. 142, no. 2, pp. 207–214, 2007. [Online]. Available: http://dx.doi.org/10.1016/j.surg.2007.04.016

[13] R. Marmulla, S. Hassfeld, T. Lüth, and J. Mühling, "Laser-scan-based navigation in cranio-maxillofacial surgery," *Journal of Cranio-Maxillofacial Surgery*, vol. 31, no. 5, pp. 267–277, 2003. [Online]. Available: http://dx.doi.org/10.1016/S1010-5182(03)00056-8

[14] R. Marmulla, G. Eggers, and J. Mühling, "Laser surface registration for lateral skull base surgery," *Minimally Invasive Neurosurgery*, vol. 48, no. 3, pp. 181–185, 2005. [Online]. Available: http://dx.doi.org/10.1055/s-2005-870906

[15] A. Cao, R. C. Thompson, P. al Dumpuri, B. M. Dawant, R. L.

Galloway, S. Ding, and M. I. Miga, "Laser range scanning for image-guided neurosurgery: Investigation of image-to-physical space registrations," *Medical Physics*, vol. 35, no. 4, pp. 1593–1605, 2008. [Online]. Available: http://dx.doi.org/10.1118/1.2870216

[16] C. Bert, K. G. Metheany, K. Doppke, and G. T. Y. Chen, "A phantom evaluation of a stereo-vision surface imaging system for radiotherapy patient setup," *Medical Physics*, vol. 32, no. 9, pp. 2753–2762, 2005. [Online]. Available: http://dx.doi.org/10.1118/1.1984263

[17] J. L. Herring, B. M. Dawant, C. R. Maurer Jr, D. M. Muratore, R. L. Galloway, and J. M. Fitzpatrick, "Surface-based registration of CT images to physical space for image-guided surgery of the spine: A sensitivity study," *IEEE Transactions on Medical Imaging*, vol. 17, no. 5, pp. 743–752, 1998. [Online]. Available: http://dx.doi.org/10.1109/42.736029

[18] D. V. Amin, T. Kanade, A. M. Digioia, and B. Jaramaz, "Ultrasound registration of the bone surface for surgical navigation," *Computer Aided Surgery*, vol. 8, no. 1, pp. 1–16, 2003. [Online]. Available: http://dx.doi.org/10.3109/10929080309146097

[19] C. A. Pelizzari, G. T. Chen, D. R. Spelbring, R. R. Weichselbaum, and C.-T. Chen, "Accurate three-dimensional registration of CT, PET, and/or MR

images of the brain," *Journal of Computer Assisted Tomography*, vol. 13, no. 1, pp. 20–26, 1989.

[20] L.-Y. Hsu and M. H. Loew, "Fully automatic 3D feature-based registration of multi-modality medical images," *Image and Vision Computing*, vol. 19, no. 1-2, pp. 75–85, Jan. 2001. [Online]. Available: http://dx.doi.org/10.1016/S0262-8856(00)00058-5

[21] W.-C. C. Lee, M. E. Tublin, and B. E. Chapman, "Registration of MR and CT images of the liver: Comparison of voxel similarity and surface based registration algorithms," *Computer Methods and Programs in Biomedicine*, vol. 78, no. 2, pp. 101–114, 2005. [Online]. Available: http://dx.doi.org/10.1016/j.cmpb.2004.12.006

[22] X. Kang, W. P. Yau, Y. Otake, P. Y. S. Cheung, Y. Hu, and R. H. Taylor, "Assessing 3D tunnel position in ACL reconstruction using a novel single image 3D-2D registration," in *SPIE, Medical Imaging 2012: Image-Guided Procedures, Robotic Interventions, and Modeling*, D. R. Holmes III and K. H. Wong, Eds., vol. 8316. International Society for Optics and Photonics, 2012, p. 831628. [Online]. Available: http://dx.doi.org/10.1117/12.911131

[23] N. Baka, C. Metz, C. Schultz, R.-J. van Geuns, W. Niessen, and T. van Walsum, "Oriented gaussian mixture models for non-rigid 2D/3D coronary artery

registration," *IEEE Transactions on Medical Imaging*, vol. 33, no. 5, pp. 1023–1034, 2014. [Online]. Available: http://dx.doi.org/10.1109/TMI.2014.2300117

[24] D. J. Mirota, W. Hanzi, R. H. Taylor, I. M, G. L. Gallia, and G. D. Hager, "A system for video-based navigation for endoscopic endonasal skull base surgery," *IEEE Transactions on Medical Imaging*, vol. 31, no. 4, pp. 963–976, 2012. [Online]. Available: http://dx.doi.org/10.1109/TMI.2011.2176500

[25] S. Billings and R. Taylor, "Iterative most likely oriented point registration," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2014*, ser. Lecture Notes in Computer Science, P. Golland, N. Hata, C. Barillot, J. Hornegger, and R. Howe, Eds. Springer International Publishing, 2014, vol. 8673, pp. 178–185. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10404-1_23

[26] S. D. Billings, E. M. Boctor, and R. H. Taylor, "Iterative most-likely point registration (IMLP): A robust algorithm for computing optimal shape alignment," *PLoS ONE*, vol. 10, no. 3, p. e0117688, 2015. [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0117688

[27] S. D. Billings and R. H. Taylor, "Generalized iterative most likely oriented-point (G-IMLOP) registration," *International Journal of Computer Assisted Radiology and Surgery*, vol. 10, no. 8, pp. 1213–1226, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11548-015-1221-2

[28] X. Liu, H. Cevikalp, and J. M. Fitzpatrick, "Marker orientation in fiducial registration," in *SPIE, Medical Imaging 2003: Image Processing*, M. Sonka and J. M. Fitzpatrick, Eds., vol. 5032. International Society for Optics and Photonics, 2003, pp. 1176–1185. [Online]. Available: http://dx.doi.org/10.1117/12.480860

[29] S. D. Billings, H. J. Kang, A. Cheng, E. M. Boctor, P. Kazanzides, and R. H. Taylor, "Minimally invasive registration for computer-assisted orthopedic surgery: Combining tracked ultrasound and bone surface points via the P-IMLOP algorithm," *International Journal of Computer Assisted Radiology and Surgery*, vol. 10, no. 6, pp. 761–771, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11548-015-1188-z

[30] H. Samet, *Foundations of multidimensional and metric data structures*. San Francisco, CA: Morgan Kaufmann, 2006.

[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes: The art of scientific computing*, 3rd ed. New York, NY: Cambridge University Press, 2007.

[32] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, Sep. 1987. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.1987.4767965

[33] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987. [Online]. Available: http://dx.doi.org/10.1364/JOSAA.4.000629

[34] M. W. Walker, L. Shao, and R. A. Volz, "Estimating 3-D location parameters using dual number quaternions," *CVGIP: image understanding*, vol. 54, no. 3, pp. 358–367, 1991. [Online]. Available: http://dx.doi.org/10.1016/1049-9660(91)90036-O

[35] G. Champleboux, S. Lavallee, R. Szeliski, and L. Brunie, "From accurate range imaging sensor calibration to accurate model-based 3D object localization," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, Jun. 1992, pp. 83–89. [Online]. Available: http://dx.doi.org/10.1109/CVPR.1992.223223

[36] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.1991.132043

[37] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994. [Online]. Available: http://dx.doi.org/10.1007/BF01427149

[38] C. R. Maurer Jr, G. B. Aboutanos, B. M. Dawant, R. J. Maciunas, and J. M. Fitzpatrick, "Registration of 3-D images using weighted geometrical features,"

IEEE Transactions on Medical Imaging, vol. 15, no. 6, pp. 836–849, 1996. [Online]. Available: http://dx.doi.org/10.1109/42.544501

[39] L. Armesto, J. Minguez, and L. Montesano, "A generalization of the metric-based iterative closest point technique for 3D scan matching," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1367–1372. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.2010.5509371

[40] J. Minguez, L. Montesano, and F. Lamiraux, "Metric-based iterative closest point scan matching for sensor displacement estimation," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 1047–1054, 2006. [Online]. Available: http://dx.doi.org/10.1109/TRO.2006.878961

[41] A. W. Fitzgibbon, "Robust registration of 2D and 3D point sets," *Image and Vision Computing*, vol. 21, no. 13-14, pp. 1145–1153, Dec. 2003. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2003.09.004

[42] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness, "New algorithms for 2D and 3D point matching: Pose estimation and correspondence," *Pattern Recognition*, vol. 31, no. 8, pp. 1019–1031, 1998. [Online]. Available: http://dx.doi.org/10.1016/S0031-3203(98)80010-1

[43] H. Chui and A. Rangarajan, "A feature registration framework using mixture models," in *Mathematical Methods in Biomedical Image Analysis,*

2000. Proceedings. IEEE Workshop on. IEEE, 2000, pp. 190–197. [Online]. Available: http://dx.doi.org/10.1109/MMBIA.2000.852377

[44] S. Granger and P. Xavier, "Multi-scale EM-ICP: A fast and robust approach for surface registration," in *Computer Vision—ECCV 2002*, ser. Lecture Notes in Computer Science, A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, Eds. Springer Berlin Heidelberg, 2002, vol. 2353, pp. 418–432. [Online]. Available: http://dx.doi.org/10.1007/3-540-47979-1_28

[45] B. Combès and S. Prima, "An efficient EM-ICP algorithm for symmetric consistent non-linear registration of point sets," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2010*, ser. Lecture Notes in Computer Science, T. Jiang, N. Navab, J. P. W. Pluim, and M. A. Viergever, Eds. Springer Berlin Heidelberg, 2010, vol. 6362, pp. 594–601. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15745-5_73

[46] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2010.46

[47] Y. Tsin and T. Kanade, "A correlation-based approach to robust point set registration," in *Computer Vision–ECCV 2004*. Springer, 2004, pp. 558–569. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24672-5_44

[48] B. Jian and B. C. Vemuri, "Robust point set registration using Gaussian mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 8, pp. 1633–1645, 2011. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2010.223

[49] N. Verma, S. Kpotufe, and S. Dasgupta, "Which spatial partition trees are adaptive to intrinsic dimension?" in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. Arlington: AUAI Press, 2009, pp. 565–574.

[50] J. P. Williams, R. H. Taylor, and L. B. Wolff, "Augmented KD techniques for accelerated registration and distance measurement of surfaces," in *Computer Aided Surgery: Computer-Integrated Surgery of the Head and Spine*, Sep. 1997, pp. 1–21.

[51] J. J. Craig, *Introduction to robotics: Mechanics and control*, 3rd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.

[52] J. T. Bushberg, J. A. Seibert, E. M. Leidholdt Jr, and J. M. Boone, *The essential physics of medical imaging*, 3rd ed. Philadelphia, PA: Lippincott Williams & Wilkins, 2012.

[53] J. Chen, Z. Ding, and F. Yuan, "Theoretical uncertainty evaluation of stereo reconstruction," in *Bioinformatics and Biomedical Engineering, 2008. ICBBE*

2008. The 2nd International Conference on. IEEE, 2008, pp. 2378–2381. [Online]. Available: http://dx.doi.org/10.1109/ICBBE.2008.927

[54] G. Di Leo, C. Liguori, and A. Paolillo, "Covariance propagation for the uncertainty estimation in stereo vision," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 5, pp. 1664–1673, 2011. [Online]. Available: http://dx.doi.org/10.1109/TIM.2011.2113070

[55] R. S. J. Estépar, A. Brun, and C.-F. Westin, "Robust generalized total least squares iterative closest point registration," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2004*, ser. Lecture Notes in Computer Science, C. Barillot, D. R. Haynor, and P. Hellier, Eds. Springer Berlin Heidelberg, 2004, vol. 3216, pp. 234–241. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30135-6_29

[56] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," in *Robotics: Science and Systems V*, J. Trinkle, Y. Matsuoka, and J. A. Castellanos, Eds. MIT Press, 2009. [Online]. Available: http://www.roboticsproceedings.org/rss05/p21.html

[57] L. Maier-Hein, A. M. Franz, T. R. Dos Santos, M. Schmidt, M. Fangerau, H. P. Meinzer, and J. M. Fitzpatrick, "Convergent iterative closest-point algorithm to accomodate anisotropic and inhomogenous localization error," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34,

no. 8, pp. 1520–1532, 2012. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2011.248

[58] M. H. Moghari and P. Abolmaesumi, "Point-based rigid-body registration using an unscented Kalman filter," *IEEE Transactions on Medical Imaging*, vol. 26, no. 12, pp. 1708–1728, 2007. [Online]. Available: http://dx.doi.org/10.1109/TMI.2007.901984

[59] F. Nazem, A. Ahmadian, N. D. Seraj, and M. Giti, "Two-stage point-based registration method between ultrasound and CT imaging of the liver based on ICP and unscented Kalman filter: A phantom study," *International journal of computer assisted radiology and surgery*, vol. 9, no. 1, pp. 39–48, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11548-013-0907-6

[60] X. Pennec and J.-P. Thirion, "A framework for uncertainty and validation of 3-D registration methods based on points and frames," *International Journal of Computer Vision*, vol. 25, no. 3, pp. 203–229, 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1007976002485

[61] R. Balachandran and J. M. Fitzpatrick, "Iterative solution for rigid-body point-based registration with anisotropic weighting," in *SPIE, Medical Imaging 2009: Visualization, Image-Guided Procedures, and Modeling*, M. I. Miga and K. H. Wong, Eds. International Society for Optics and Photonics, 2009, p. 72613D. [Online]. Available: http://dx.doi.org/10.1117/12.813887

[62] N. Ohta and K. Kanatani, "Optimal estimation of three-dimensional rotation and reliability evaluation," *IEICE Transactions on Information and Systems*, vol. 81, no. 11, pp. 1247–1252, 1998.

[63] K. Kanatani, *Statistical optimization for geometric computation: Theory and practice*. Mineola, NY: Dover Publications, Inc., 1996.

[64] B. C. Matei and P. Meer, "Estimation of nonlinear errors-in-variables models for computer vision applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1537–1552, 2006. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2006.205

[65] A. Danilchenko and J. M. Fitzpatrick, "General approach to first-order error prediction in rigid point registration," *IEEE Transactions on Medical Imaging*, vol. 30, no. 3, pp. 679–693, 2011. [Online]. Available: http://dx.doi.org/10.1109/TMI.2010.2091513

[66] J. Rice, *Mathematical statistics and data analysis*, 3rd ed. Belmont, CA: Brooks/Cole, 2007.

[67] T. Larsson, "An efficient ellipsoid-OBB intersection test," *Journal of Graphics, GPU, and Game Tools*, vol. 13, no. 1, pp. 31–43, 2008. [Online]. Available: http://dx.doi.org/10.1080/2151237X.2008.10129253

[68] M. Fiedler, "Bounds for the determinant of the sum of Hermitian matrices,"

Proceedings of the American Mathematical Society, vol. 30, no. 1, pp. 27–31, 1971. [Online]. Available: http://dx.doi.org/10.2307/2038212

[69] E. A. Yildirim, "On the minimum volume covering ellipsoid of ellipsoids," *SIAM Journal on Optimization*, vol. 17, no. 3, pp. 621–641, 2006. [Online]. Available: http://dx.doi.org/10.1137/050622560

[70] P. Gans, *Data fitting in the chemical sciences: By the method of least squares*. Hoboken, NJ: Wiley, 1992.

[71] W. E. Wentworth, "Rigorous least squares adjustment: Application to some non-linear equations, I," *Journal of Chemical Education*, vol. 42, no. 2, pp. 96–103, 1965. [Online]. Available: http://dx.doi.org/10.1021/ed042p96

[72] A. Deguet, R. Kumar, R. Taylor, and P. Kazanzides, "The CISST libraries for computer assisted intervention systems," *MIDAS Journal - Systems and Architectures for Computer Assisted Interventions (MICCAI 2008 Workshop)*, pp. 1–8, 2008. [Online]. Available: http://hdl.handle.net/10380/1465

[73] D. H. Eberly, *Game physics*, 2nd ed. Boca Raton, FL: CRC Press, 2010.

[74] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: An open-source mesh processing tool," in *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008, pp.

129–136. [Online]. Available: http://dx.doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136

[75] T. Tamaki, M. Abe, B. Raytchev, and K. Kaneda, "Softassign and EM-ICP on GPU," in *Networking and Computing (ICNC), 2010 First International Conference on*. IEEE, 2010, pp. 179–183. [Online]. Available: http://dx.doi.org/10.1109/IC-NC.2010.60

[76] T. Tsujimura and T. Yabuta, "A tactile sensing method for employing force/torque information through insensitive probes," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, vol. 2, 1992, pp. 1315–1320. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.1992.220167

[77] M. Charlebois, K. Gupta, and S. Payandeh, "Shape description of general, curved surfaces using tactile sensing and surface normal information," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 4. IEEE, 1997, pp. 2819–2824. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.1997.606714

[78] H. Maekawa, K. Tanie, and K. Komoriya, "A finger-shaped tactile sensor using an optical waveguide," in *Systems, Man and Cybernetics, 1993.'Systems Engineering in the Service of Humans', Conference Proceedings., International*

Conference on, vol. 5. IEEE, 1993, pp. 403–408. [Online]. Available: http://dx.doi.org/10.1109/ICSMC.1993.390885

[79] Z. Song and R. Chung, "Nonstructured light-based sensing for 3D reconstruction," *Pattern Recognition*, vol. 43, no. 10, pp. 3560–3571, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2010.05.008

[80] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi, "Efficiently combining positions and normals for precise 3D geometry," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 536–543, 2005. [Online]. Available: http://dx.doi.org/10.1145/1073204.1073226

[81] T. Lange, S. Eulenstein, M. Hünerbein, and P.-M. Schlag, "Vessel-based non-rigid registration of MR/CT and 3D ultrasound for navigation in liver surgery," *Computer Aided Surgery*, vol. 8, no. 5, pp. 228–240, 2003. [Online]. Available: http://dx.doi.org/10.3109/10929080309146058

[82] K. Pulli, "Multiview registration for large data sets," in *Second International Conference on 3-D Digital Imaging and Modeling, 1999*, 1999, pp. 160–168. [Online]. Available: http://dx.doi.org/10.1109/IM.1999.805346

[83] C. Lara, L. Romero, and F. Calderón, "A robust iterative closest point algorithm with augmented features," in *MICAI 2008: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, A. Gelbukh and E. F.

Morales, Eds. Springer Berlin Heidelberg, 2008, vol. 5317, pp. 605–614. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88636-5_58

[84] D. Münch, B. Combès, and S. Prima, "A modified ICP algorithm for normal-guided surface registration," in *SPIE, Medical Imaging 2010: Image Processing*, B. M. Dawant and D. R. Haynor, Eds., vol. 7623, 2010, p. 76231A. [Online]. Available: http://dx.doi.org/10.1117/12.844994

[85] K. V. Mardia and P. E. Jupp, *Directional statistics*, ser. Wiley Series in Probability and Statistics. West Sussex, England: John Wiley & Sons, Ltd., 2000.

[86] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Sra, and G. Ridgeway, "Clustering on the unit hypersphere using von Mises-Fisher distributions," *Journal of Machine Learning Research*, vol. 6, no. 9, pp. 1345–1382, 2005.

[87] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[88] B. Preising, T. C. Hsia, and B. Mittelstadt, "A literature review: Robots in medicine," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 10, no. 2, pp. 13–22, Jun. 1991. [Online]. Available: http://dx.doi.org/10.1109/51.82001

[89] H. A. Paul, W. L. Bargar, B. Mittelstadt, B. Musits, R. H. Taylor, P. Kazanzides, J. Zuhars, B. Williamson, and W. Hanson, "Development of

a surgical robot for cementless total hip arthroplasty," *Clinical Orthopaedics and Related Research*, vol. 285, pp. 57–66, Dec. 1992.

[90] R. H. Taylor, B. D. Mittelstadt, H. A. Paul, W. Hanson, P. Kazanzides, J. F. Zuhars, B. Williamson, B. L. Musits, E. Glassman, and W. L. Bargar, "An image-directed robotic system for precise orthopaedic surgery," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 261–275, Jun. 1994. [Online]. Available: http://dx.doi.org/10.1109/70.294202

[91] P. Kazanzides, J. Zuhars, B. Mittelstadt, and R. H. Taylor, "Force sensing and control for a surgical robot," in *IEEE International Conference on Robotics and Automation*, vol. 1, Nice, France, May 1992, pp. 612–617. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.1992.220224

[92] M. Nogler, H. Maurer, C. Wimmer, C. Gegenhuber, C. Bach, and M. Krismer, "Knee pain caused by a fiducial marker in the medial femoral condyle: A clinical and anatomic study of 20 cases," *Acta Orthopaedica*, vol. 72, no. 5, pp. 477–480, Oct. 2001. [Online]. Available: http://dx.doi.org/10.1080/000164701753532808

[93] A. Guéziec, P. Kazanzides, B. Williamson, and R. H. Taylor, "Anatomy-based registration of CT-scan and intraoperative X-ray images for guiding a surgical robot," *IEEE Transactions on Medical Imaging*, vol. 17, no. 5, pp. 715–728, 1998. [Online]. Available: http://dx.doi.org/10.1109/42.736023

[94] D. LaRose, J. Bayouth, and T. Kanade, "Transgraph: Interactive intensity-

based 2D/3D registration of X-ray and CT data," in *SPIE, Medical Imaging 2000: Image Processing*, K. M. Hanson, Ed., vol. 3979. International Society for Optics and Photonics, 2000, pp. 385–396. [Online]. Available: http://dx.doi.org/10.1117/12.387700

[95] Y. Otake, M. Armand, R. S. Armiger, M. D. Kutzer, E. Basafa, P. Kazanzides, and R. H. Taylor, "Intraoperative image-based multiview 2D/3D registration for image-guided orthopaedic surgery: Incorporation of fiducial-based C-arm tracking and GPU-acceleration," *IEEE Transactions on Medical Imaging*, vol. 31, no. 4, pp. 948–962, 2012. [Online]. Available: http://dx.doi.org/10.1109/TMI.2011.2176555

[96] A. Sahay, L. W. Witherspoon, and W. L. Bargar, "Computer model-based study for minimally invasive THR femoral cavity preparation using the ROBODOC system," in *4th Annual Meeting, Computer Assisted Orthopaedic Surgery (CAOS)*, Chicago, IL, Jun. 2004, pp. 314–316.

[97] C. R. Maurer, R. P. Gaston, D. L. G. Hill, M. J. Gleeson, M. G. Taylor, M. R. Fenlon, P. J. Edwards, and D. J. Hawkes, "AcouStick: A tracked A-mode ultrasonography system for registration in image-guided surgery," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI'99*, ser. Lecture Notes in Computer Science, C. Taylor and A. Colchester, Eds.,

vol. 1679. Springer Berlin Heidelberg, 1999, pp. 953–962. [Online]. Available: http://dx.doi.org/10.1007/10704282_104

[98] T. K. Chen, P. Abolmaesumi, D. R. Pichora, and R. E. Ellis, "A system for ultrasound-guided computer-assisted orthopaedic surgery," *Computer Aided Surgery*, vol. 10, no. 5-6, pp. 281–292, 2005. [Online]. Available: http://dx.doi.org/10.3109/10929080500390017

[99] D. C. Barratt, G. P. Penney, C. S. K. Chan, M. Slomczykowski, T. J. Carter, P. J. Edwards, and D. J. Hawkes, "Self-calibrating 3D-ultrasound-based bone registration for minimally invasive orthopedic surgery," *IEEE Transactions on Medical Imaging*, vol. 25, no. 3, pp. 312–323, Mar. 2006. [Online]. Available: http://dx.doi.org/10.1109/TMI.2005.862736

[100] P. Foroughi, E. Boctor, M. J. Swartz, R. H. Taylor, and G. Fichtinger, "Ultrasound bone segmentation using dynamic programming," in *IEEE Ultrasonics Symposium*, Oct. 2007, pp. 2523–2526. [Online]. Available: http://dx.doi.org/10.1109/ULTSYM.2007.635

[101] J. Kowal, C. Amstutz, F. Langlotz, H. Talib, and M. G. Ballester, "Automated bone contour detection in ultrasound B-mode images for minimally invasive registration in computer-assisted surgery–an in vitro evaluation," *International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 3, no. 4, pp. 341–348, 2007. [Online]. Available: http://dx.doi.org/10.1002/rcs.160

[102] I. Hacihaliloglu, D. R. Wilson, M. Gilbart, M. A. Hunt, and P. Abolmaesumi, "Non-iterative partial view 3D ultrasound to CT registration in ultrasound-guided computer-assisted orthopedic surgery," *International Journal of Computer Assisted Radiology and Surgery*, vol. 8, no. 2, pp. 157–168, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11548-012-0747-9

[103] P. J. S. Gonçalves, P. M. B. Torres, F. Santos, R. António, N. Catarino, and J. M. M. Martins, "A vision system for robotic ultrasound guided orthopaedic surgery," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 2, pp. 327–339, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10846-013-0012-7

[104] A. K. Jain and R. H. Taylor, "Understanding bone responses in B-mode ultrasound images and automatic bone surface extraction using a Bayesian probabilistic framework," in *SPIE, Medical Imaging 2004: Ultrasonic Imaging and Signal Processing*, W. F. Walker and S. Y. Emelianov, Eds., vol. 5373, San Diego, CA, Feb. 2004, pp. 131–142. [Online]. Available: http://dx.doi.org/10.1117/12.535984

[105] V. Daanen, J. Tonetti, and J. Troccaz, "A fully automated method for the delineation of osseous interface in ultrasound images," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2004*, ser. Lecture Notes in Computer Science, C. Barillot, D. R. Haynor, and P. Hellier, Eds.,

vol. 3216. Springer Berlin Heidelberg, 2004, pp. 549–557. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30135-6_67

[106] X. Guo, A. Cheng, H. K. Zhang, H.-J. Kang, R. Etienne-Cummings, and E. M. Boctor, "Active echo: A new paradigm for ultrasound calibration," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2014*, ser. Lecture Notes in Computer Science, P. Golland, N. Hata, C. Barillot, J. Hornegger, and R. Howe, Eds. Springer International Publishing, 2014, vol. 8674, pp. 397–404. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10470-6_50

[107] A. Cheng, X. Guo, H. K. Zhang, H. J. Kang, R. Etienne-Cummings, and E. M. Boctor, "Active point out-of-plane ultrasound calibration," in *SPIE, Medical Imaging 2015: Image-Guided Procedures, Robotic Interventions, and Modeling*, Z. R. Yaniv and R. J. Webster, Eds., vol. 9415, Feb. 2015, p. 94150W. [Online]. Available: http://dx.doi.org/10.1117/12.2081662

[108] *ROBODOC user manual*, Integrated Surgical Systems, 1999.

[109] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[110] A. Leonardis, H. Bischof, and A. Pinz, "SURF: Speeded up robust features," in *Computer Vision ECCV 2006*, ser. Lecture Notes in Computer

Science, A. Leonardis, H. Bischof, and A. Pinz, Eds., vol. 3951. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. [Online]. Available: http://dx.doi.org/10.1007/11744023_32

[111] E. Delponte, F. Isgrò, F. Odone, and A. Verri, "SVD-matching using SIFT features," *Graphical Models*, vol. 68, no. 5-6, pp. 415–431, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.gmod.2006.07.002

[112] H. Wang, D. Mirota, G. Hager, and M. Ishii, "Anatomical reconstruction from endoscopic images: Toward quantitative endoscopy," *American Journal of Rhinology*, vol. 22, no. 1, pp. 47–51, Jan. 2008. [Online]. Available: http://dx.doi.org/10.2500/ajr.2008.22.3129

[113] P. H. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Vision Algorithms: Theory and Practice - International Workshop on Vision Algorithms Corfu, Greece, September 21-22, 1999 Proceedings*, ser. Lecture Notes in Computer Science, B. Triggs, A. Zisserman, and R. Szeliski, Eds., vol. 1883. Corfu, Greece: Springer-Verlag Berlin Heidelberg, 2000, pp. 278–294. [Online]. Available: http://dx.doi.org/10.1007/3-540-44480-7_19

[114] T. Huang and A. Netravali, "Motion and structure from feature correspondences: A review," *Proceedings of the IEEE*, vol. 82, no. 2, pp. 252–268, 1994. [Online]. Available: http://dx.doi.org/10.1109/5.265351

[115] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981. [Online]. Available: http://dx.doi.org/10.1038/293133a0

[116] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An invitation to 3-D vision: From images to geometric models*. New York: Springer-Verlag, 2004.

[117] D. J. Mirota, "Video-based navigation with application to endoscopic skull base surgery," Ph.D. dissertation, Johns Hopkins University, 2012.

[118] C. H. Snyderman, H. Pant, R. L. Carrau, D. Prevedello, P. Gardner, and A. B. Kassam, "What are the limits of endoscopic sinus surgery?: The expanded endonasal approach to the skull base," *The Keio Journal of Medicine*, vol. 58, no. 3, pp. 152–160, Oct. 2009. [Online]. Available: http://dx.doi.org/10.2302/kjm.58.152

[119] A. B. Kassam, D. M. Prevedello, R. L. Carrau, C. H. Snyderman, A. Thomas, P. Gardner, A. Zanation, B. Duz, S. T. Stefko, K. Byers, and M. B. Horowitz, "Endoscopic endonasal skull base surgery: Analysis of complications in the authors' initial 800 patients," *Journal of neurosurgery*, vol. 114, no. 6, pp. 1544–1568, Jun. 2011. [Online]. Available: http://dx.doi.org/10.3171/2010.10.JNS09406

[120] N. A. Cohen and D. W. Kennedy, "Endoscopic sinus surgery: Where we are-

and where we're going," *Current Opinion in Otolaryngology & Head and Neck Surgery*, vol. 13, no. 1, pp. 32–38, 2005.

[121] R. Metson, R. E. Gliklich, and M. Cosenza, "A comparison of image guidance systems for sinus surgery," *The Laryngoscope*, vol. 108, no. 8, pp. 1164–1170, Aug. 1998. [Online]. Available: http://dx.doi.org/10.1097/00005537-199808000-00012

[122] J. B. Anon, "Computer-aided endoscopic sinus surgery," *The Laryngoscope*, vol. 108, no. 7, pp. 949–961, Jul. 1998. [Online]. Available: http://dx.doi.org/10.1097/00005537-199807000-00001

[123] M. P. Fried, V. M. Moharir, J. Shin, M. Taylor-Becker, and P. Morrison, "Comparison of endoscopic sinus surgery with and without image guidance," *American Journal of Rhinology*, vol. 16, no. 4, pp. 193–197, 2002.

[124] K. Cleary and T. M. Peters, "Image-guided interventions: Technology review and clinical applications," *Annual Review of Biomedical Engineering*, vol. 12, pp. 119–142, Aug. 2010. [Online]. Available: http://dx.doi.org/10.1146/annurev-bioeng-070909-105249

[125] D. J. Mirota, M. Ishii, and G. D. Hager, "Vision-based navigation in image-guided interventions," *Annual Review of Biomedical Engineering*, vol. 13, pp. 297–319, Aug. 2011. [Online]. Available: http://dx.doi.org/10.1146/annurev-bioeng-071910-124757

[126] D. Burschka, M. Li, M. Ishii, R. H. Taylor, and G. D. Hager, "Scale-invariant registration of monocular endoscopic images to CT-scans for sinus surgery," *Medical Image Analysis*, vol. 9, no. 5, pp. 413–426, Oct. 2005. [Online]. Available: http://dx.doi.org/10.1016/j.media.2005.05.005

[127] D. Mirota, R. H. Taylor, M. Ishii, and G. D. Hager, "Direct endoscopic video registration for sinus surgery," in *SPIE, Medical Imaging 2009: Visualization, Image-Guided Procedures, and Modeling*, M. I. Miga and K. H. Wong, Eds. International Society for Optics and Photonics, Feb. 2009, p. 72612K. [Online]. Available: http://dx.doi.org/10.1117/12.812334

[128] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *Object recognition supported by user interaction for service robots*, vol. 3. IEEE, 2002, pp. 545–548. [Online]. Available: http://dx.doi.org/10.1109/ICPR.2002.1047997

[129] D. Mirota, H. Wang, R. H. Taylor, M. Ishii, and G. D. Hager, "Toward video-based navigation for endoscopic endonasal skull base surgery," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2009*, G.-Z. Yang, D. Hawkes, D. Rueckert, A. Noble, and C. Taylor, Eds., vol. 5761. Springer Berlin Heidelberg, Jan. 2009, pp. 91–99. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04268-3_12

[130] E. Lengyel, *Mathematics for 3D game programming and computer graphics*, 3rd ed. Boston, MA: Cengage Learning, 2012.

[131] R. Davies, C. Twining, and C. Taylor, *Statistical models of shape: Optimisation and evaluation*. London: Springer-Verlag, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-1-84800-138-1

[132] E. Freeman, E. Robson, B. Bates, and K. Sierra, *Head first design patterns*. Sebastopol, CA: O'Reilly Media, 2004.

[133] T. Kailath, *Linear systems*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1980.

[134] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge, UK: Cambridge University Press, 1999.

Vita

Seth Billings was born in Virginia, Minnesota on August 23, 1984. He grew up in the small village of Pellston, Michigan, completing a homeschool-based high school education in 2003. He received the Bachelor's of Science Degree in Electrical Engineering from Kettering University in 2007, graduating summa cum laude with dual minors in Computer Engineering and Applied Optics. In 2008, he enrolled in the Computer Science Ph.D. program at Johns Hopkins University and also received the Master's of Science in Engineering Degree from Johns Hopkins University in 2010. His graduate research at Johns Hopkins University has focused on systems and probabilistic registration algorithms for computer-integrated medical interventions. Throughout his undergraduate career, he worked as a cooperative employment engineer at Dematic Corporation, which culminated in the embedded systems topic of his bachelor's thesis, which he completed with distinction. Following graduation, he worked full-time as a Product Engineer in the Research & Development division of Dematic Corporation prior to enrolling at Johns Hopkins University. He is a fellow of the National Science Foundation Graduate Research Program Fellowship awarded

in 2010 and was a fellow of the National Institutes of Health Individual Graduate Program Partnership in 2010–2012. He holds membership in numerous honor societies, including Tau Beta Pi, Eta Kappa Nu, Kappa Mu Epsilon, and Phi Eta Sigma, and has received numerous academic awards, such as the Rob Roy Graduate Scholarship, Kettering Presidential Scholarship, MICCAI 2014 Student Travel Award, and Kettering University Freshman Student of the Year Award.